Adaptable Transformation-Based Similarity Search in Multimedia Databases

Von der Fakultät für Mathematik, Informatik und Naturwissenschaften der RWTH Aachen University zur Erlangung des akademischen Grades eines Doktors der Naturwissenschaften genehmigte Dissertation

vorgelegt von

Diplom-Informatiker
Marc Wichterich
aus Haan

Berichter: Universitätsprofessor Dr. rer. nat. Thomas Seidl
Universitätsprofessor Dr. rer. nat. Andreas Henrich

Tag der mündlichen Prüfung: 12.10.2010

Diese Dissertation ist auf den Internetseiten der Hochschulbibliothek online verfügbar.
To those who supported me during the past five years.
Contents

Abstract / Zusammenfassung 1

I Introduction to Distance-Based Similarity Search 5

1 Distance-Based Similarity Model 9
  1.1 Object Feature Representations 9
  1.2 Distance Measures 14

2 Distance-Based Similarity Queries 19
  2.1 Range Queries 19
  2.2 Nearest Neighbor Queries 21
  2.3 Ranking Queries 22

3 Efficient Similarity Query Processing 25
  3.1 Lower Bounds and Multi-Step Processing 25
  3.2 Indexing Structures 27

4 An Example for Similarity Search Using Simple Distance Measures 31
  4.1 Introduction 31
  4.2 Preliminaries 34
  4.3 Dimensionality Reduction for the Voronoi Approach to NN Search 38
  4.4 Experiments 51
  4.5 Summary 56
## II Transformation-Based Distance Measures

### 5 The Edit Distance

- 5.1 Formal Definition ................................................................. 61
- 5.2 Illustration of Core Concepts of Transformation-Based Distances 63
- 5.3 Computation ................................................................. 64

### 6 The Dynamic Time Warping Distance

- 6.1 Introduction ................................................................. 67
- 6.2 Formal Definition ................................................................. 69
- 6.3 Computation ................................................................. 71

### 7 The Earth Mover’s Distance

- 7.1 Introduction ................................................................. 75
- 7.2 Applications ................................................................. 79
- 7.3 Formal Definition ................................................................. 80
- 7.4 Computation ................................................................. 84
- 7.5 Approximations of the EMD .................................................. 86

## III Fast Searching with the Earth Mover’s Distance

### 8 Efficient Search Via Flexible Dimensionality Reduction

- 8.1 Introduction ................................................................. 95
- 8.2 Dimensionality Reduction for the EMD ...................................... 97
- 8.3 Query Processing Algorithm .................................................. 113
- 8.4 Experiments ................................................................. 116
- 8.5 Summary ................................................................. 126

### 9 Direct Indexing and Vector Approximation

- 9.1 Introduction ................................................................. 127
- 9.2 Direct Indexing of the EMD .................................................. 129
- 9.3 Vector Approximation for the EMD ......................................... 133
- 9.4 Experiments ................................................................. 143
- 9.5 Summary ................................................................. 148
IV Effective Searching with the Earth Mover’s Distance 149

10 Metric Adaptation of the Earth Mover’s Distance 153
10.1 Introduction ................................................. 153
10.2 Mathematical Framework for Adapting a Metric EMD ........ 154
10.3 Summary ..................................................... 166

11 Exploring Multimedia Databases via Relevance Feedback 169
11.1 Introduction .................................................. 169
11.2 Relevance Feedback with the Earth Mover’s Distance .......... 172
11.3 Experiments .................................................. 188
11.4 Summary ..................................................... 196

V The Earth Mover’s Distance in Context of Structured Objects 197

12 Feature-Based Graph Similarity with the Earth Mover’s Distance 201
12.1 Introduction .................................................. 201
12.2 Preliminaries ................................................ 203
12.3 Graph Similarity Model ................................... 204
12.4 Experimental Evaluation ................................... 209
12.5 Summary and Outlook ................................... 213

13 Reducing the Number of DTW Ground Distance Computations 215
13.1 Introduction .................................................. 215
13.2 Anticipatory DTW ......................................... 218
13.3 Experiments ................................................ 232
13.4 Summary and Outlook ................................... 239
VI  Summary and Outlook  241

VII  Appendices  I
   Naming Conventions  III
   List of Symbols  V
   List of Algorithms  VII
   Bibliography  IX
   Collaboration / Acknowledgments  XXIX
   List of Publications  XXXIII
   Curriculum Vitae  XXXV
Abstract

Efficient and effective methods of making data accessible to its consumers – be they humans or algorithms – are crucial for turning ever-growing data dumps into data mines.

Of particular importance to the user are access methods that allow for query-based searching of databases. However, for vast collections of complex data objects such as digital image libraries and music databases, querying methods that necessitate an accurate, algebraic description of what the user is looking for cannot cover all search needs. For instance, a prototypical object might be known to the user and yet he or she may be unable to describe which qualities make the object prototypical. Similarity search systems based on the query-by-example paradigm can help the user in such situations by retrieving objects from the database that exhibit a high degree of similarity to the prototypical query object. For this purpose, the system must decide algorithmically which objects are to be deemed similar to each other.

After giving an introduction and reviewing preliminaries in parts I and II, the following three parts of this thesis address novel techniques regarding the efficiency, effectiveness, and applicability of a particularly intuitive and flexible class of distance measures where the distance (i.e., dissimilarity) between two objects is modeled as the minimum amount of work that is required for transforming the feature representation of one object into the feature representation of the other. As the cost of transforming a feature into another can be chosen depending on the features at hand, these transformation-based distance measures are highly adaptable and do not assume that the underlying features are perceptually independent.
Zusammenfassung


Nach einer Einleitung und einem Überblick über die Grundlagen der Arbeit in den Teilen I und II, zeigen die darauf folgenden drei Teile Techniken auf, welche die Effizienz, die Effektivität und die Anwendbarkeit einer besonders flexiblen und intuitiven Klasse von Distanzmaßen betreffen. Die Distanz zweier Objekte wird hier als Maß für die Unähnlichkeit dieser interpretiert und als minimales Pensum an Arbeit modelliert, welches für die Umwandlung der Merkmalsrepräsentation des einen Objekts in die des anderen aufzuwenden ist. Da die Kosten der für die Transformation aufzuwendenden Arbeit den auftretenden Objektmerkmalen entsprechend gewählt werden können, sind transformationsbasierte Distanzmaße in höchstem Maße anpassbar und vermögen es wahrnehmungsbezogene Abhängigkeiten der Merkmale zu berücksichtigen.
Part I

Introduction to Distance-Based Similarity Search
Ways of making data accessible to its consumers – be they humans or algorithms – are crucial for turning ever-growing data dumps into data mines.

Of particular importance to the user are access methods that allow for query-based searching of databases. For vast collections of complex data objects such as large text document corpora, digital image libraries, and music databases, querying methods that necessitate an accurate, algebraic description of what the user is looking for cannot cover all search needs. The user may only have a vague idea of the types of objects he or she would like to retrieve from the database. A prototypical object might be known to the user and yet he or she may be unable to describe, in an algebraic way, which qualities make the object prototypical. Similarity search systems are employed in such situations to help the user query the database. Given a rough sketch or an example of a prototypical query object, the task of a similarity search system is to return objects from the database that exhibit a high degree of similarity to the query object. For this purpose, the similarity search system typically establishes the degree of similarity between the query object and the database objects based on a measure of similarity that compares characteristic features of the objects.

Having a computer-evaluable measure of similarity is also of great importance in a number of algorithms and applications beyond query-driven retrieval. In the field of data mining, data clustering aims at grouping objects according to their similarity while the classification of data can be accomplished by assigning class labels to objects according to their similarity to objects with known class labels. In computer vision, image registration is often based on finding regions within an image that are similar to subimages or templates of a reference image.

The utility of a similarity measure greatly depends on the degree with which it can match the user’s notion of similarity (effectiveness) and on the speed with which search algorithms can find similar objects in the database (efficiency). If the system returns objects that the user would not judge as relevant or if it returns relevant objects but only after an excessive amount of time, the result is of little use.

In distance-based similarity search, the similarity of two objects is described by a distance function between representations of their features. A low distance value stands for a high similarity while a large distance value indicates that the
objects are dissimilar. Finding the most similar objects thus turns into finding the closest objects in distance-based similarity search.

Parts III through V of this work address novel techniques regarding the efficiency, effectiveness and applicability of a particularly intuitive and flexible class of distance functions where the distance between two objects is modeled as the amount of work that is required for transforming the features of one object into the features of the other. As the cost of transforming a feature into another feature can be chosen depending on the features at hand, these transformation-based distance measures are highly adaptable and do not assume that the underlying features are mutually independent. The distance measures discussed have been shown to be effective in a large number of applications as detailed in Part II, where several transformation-based distance measures are introduced.
Chapter 1

Distance-Based Similarity Model

Distance-based similarity models for searching within multimedia databases typically consist of two parts: An object representation and a distance measure. Multimedia databases often describe the objects they contain via the distribution of features that the objects exhibit. The selection, extraction, and representation of object features determines which attributes are to be considered relevant for the notion of similarity that is to be reflected by the distance measure. The selection and extraction of object features is largely application-dependent and shall not be the focus of this work. Exemplified with a simple feature extraction method, Section 1.1 formally defines the object representations encountered throughout the main part of this work. Section 1.2 discusses relevant properties of distance measures and gives examples of commonly encountered distance measures that are relevant to the remainder of this work.

1.1 Object Feature Representations

While pairwise distances between all objects in a database could theoretically be known \textit{a priori} (e.g., given by domain experts), this information does not suffice to answer similarity queries in the query-by-example framework, where a database is searched for objects that are similar to a user-given sample object that is not necessarily included in the database. The distance from the sample object to the database objects has to be computed based on a representation of
the features of the objects. This section formally introduces representations and notations that are of importance for the main parts of this work.

1.1.1 Representations of Feature Distributions

In this work, it is assumed that each object of a multimedia database at hand can be described in terms of a set of features from some feature space. For the example case of image objects, possible feature spaces are 3-dimensional color spaces such as RGB, HSV, or CIE Lab. These feature spaces could also be enriched with location information resulting in a 5-dimensional feature space such as XYHSV. A simple feature extraction method then is to measure the location and color information of (a sample of) the pixels of an image. The process is depicted in Figure 1.1 for the RGB feature space and a toy sample of four pixels. Other types of multimedia objects require different feature spaces.

Given a set of feature vectors in the feature space, feature histograms are compact, approximate representations of the feature distribution*

**Definition 1.1 (Fixed-Binning Feature Histogram)**

*In the literature, a histogram in vector form is sometimes called a feature vector. In this work, a feature vector lives in a feature space such as RGB. A histogram in vector form is a feature representation vector and lives in a feature representation space.

Given an object \( o \) and a disjoint partitioning \( P_1, ..., P_n \) of a feature space \( FS \), the fixed-binning feature histogram \( h^o \) for object features \( f_1^o, ..., f_m^o \) from \( FS \) is defined
as a vector \( h^o \) in \( \mathbb{R}^n \) with

\[
h^o[j] = \frac{1}{m} \sum_{i=1}^{m} \begin{cases} 1 & \text{if } f^o_i \in P_j, \\ 0 & \text{otherwise.} \end{cases} \quad \forall 1 \leq j \leq n.
\]

Notation: Where clear from context, \( o \) is used instead of \( h^o \) and \( |h^o|_\Sigma = \sum_{i=1}^{n} o[i] \).

The value \( h^o[j] \) stands for the relative frequency of features from a partition \( P_j \) of the feature space. As the term fixed-binning indicates, the partitioning of the feature space is determined only once (e.g., data-independently or by clustering feature vectors from a sample of the database) and used for the whole database and all queries. This fixed binning leads to efficiency advantages when comparing two histograms in later chapters but can lead to undesirable quantization effects. As the binning is not tailored to any single image, there might not be a bin that represents the pink parts of the example image in Figure 1.1 sufficiently well while at the same time there might be several bins for colors such as turquoise and yellow that do not have to be differentiated for the example image. Adaptive-binning histograms counter this quantization effect by tailoring the partitioning to the features of each object.

**Definition 1.2 (Adaptive-Binning Feature Histogram)**

Given an object \( o \) and a disjoint partitioning \( P^o_1, \ldots, P^o_n \) of a feature space \( FS \), the adaptive-binning feature histogram for object features \( f^o_1, \ldots, f^o_m \) from \( FS \) is defined as a set \( s^o = \{(r^o_1, w^o_1), \ldots, (r^o_n, w^o_n)\} \) of \( n \) tuples from \( FS \times \mathbb{R} \) with feature weights

\[
w^o_j = \frac{1}{m} \sum_{i=1}^{m} \begin{cases} 1 & \text{if } f^o_i \in P^o_j, \\ 0 & \text{otherwise.} \end{cases} \quad \forall 1 \leq j \leq n
\]

and \( r^o_j \) as some representative vector for partition \( P^o_j \).

Notation: Where clear from context, \( o \) is used instead of \( s^o \) and \( |s^o|_\Sigma = \sum_{i=1}^{n} w^o_i \).

With the partitioning being potentially different for each adaptive-binning histogram, its semantics have to be stored for each histogram. In Definition 1.2,
the bin representatives $r_j^o$ are intended to fulfill this purpose. A method for determining the partitioning of $FS$ is to cluster the feature vectors $f_j^o$ and to determine $r_j^o$ as the centroid of the $j^{th}$ cluster. For the example image in Figure 1.1, the large homogeneous pink area would assure that this color is well-represented in the adaptive-binning histogram even for relatively small $n$.

The ability of the adaptive approach to tailor its partitioning and representative vectors to the feature distribution of each object is exemplified by Figure 1.2. Figure 1.2 (c) shows a 15-color reconstruction of the image in Figure 1.2 (b). Each pixel is colored according to its corresponding representative vector $r_j^o$ that has been determined via a k-means clustering of the color features of the original image. Several shades of green, gray, and blue are differentiated along with a pink bin that represents the paint of the building. Figure 1.2 (f) shows the same kind of reconstruction for the image in Figure 1.2 (e). As the image in Figure 1.2 (e) has considerably different colors than the one in (b), the partitions of the feature space (and thus the representative vectors) are also different. Multiple shades of brown and pale turquoise are differentiated in this case. A fixed-binning histogram cannot tailor its feature space partitioning to
1.1. Object Feature Representations

each object in the database. Figures 1.2 (a) and (d) show according reconstructions for a feature space partitioning that was derived from a clustering of the features of a database with \( \sim 500 \) images. The fixed-binning partitioning of the feature space lacks colors that are necessary for a truthful reconstruction of the two images. A considerably larger number of bins would be required to achieve a level of quality that matches the one of the reconstruction based on the adaptive-binning histograms.

1.1.2 Feature Sequences

The order of the bins or dimensions of feature histograms are arbitrarily (but consistently) chosen for the fixed-binning approach and without a predefined order for the adaptive-binning approach. For some types of multimedia objects, there is an inherent order for one of the dimensions that plays an important role for comparing objects. A good example are video sequences, which can be modeled as a time-series of images or frames. The order of the frames is highly important when describing the video sequence as changing the order changes the semantics of the video sequence. Further examples of objects with an order-equipped dimension that is of high importance for the understanding of the data that they represent include stock data and other recurring time-based measurements.

Definition 1.3 (Feature Sequences)

A feature sequence \( t^o \) for an object \( o \) with \( m \) ordered feature descriptions is a sequence

\[
\langle t^o_1, \ldots, t^o_m \rangle
\]

where \( t^o_i \) represents the \( i^{th} \) feature description of \( o \).

Definition 1.3 is kept general in that \( t^o_i \) can be any feature description including a scalar value or an adaptive feature histogram. Its main purpose is to make the importance of the ordered dimension explicit and to allow for convenient access to the feature descriptions in Chapter 13, which focuses on time series.
1.2 Distance Measures

In distance-based similarity search, the degree of (dis)similarity between objects is given by a distance function. In its most general form, a distance function is defined as any function that assigns a non-negative value to a pair of objects.

**Definition 1.4 (Distance Function)**

Given a set $S$, a distance function $d : S \times S \rightarrow \mathbb{R}^+$ assigns to each pair of objects from $S$ a non-negative distance value.

Throughout the remainder of this work, distance functions are defined with $S$ as a set of object features or of object feature representations according to Chapter 1.1.

1.2.1 Properties of Distance Measures

In a number of scenarios, certain properties of a distance function are of importance either for the processing of queries or for the kind of similarity they can represent. The most widely regarded properties are the metric properties.

**Definition 1.5 (Metric Distance Function)**

A distance function $d$ on $S$ is called metric if and only if it has the following three properties:

1. **definiteness**: $\forall s, t \in S : d(s, t) = 0 \iff s = t$.

2. **symmetry**: $\forall s, t \in S : d(s, t) = d(t, s)$.

3. **triangle inequality**: $\forall s, t, u \in S : d(s, t) \leq d(s, u) + d(u, t)$.

The three metric properties influence the notion of (dis)similarity that a distance function can represent.

The **definiteness** property ensures that all objects have the lowest possible distance and thus the highest possible similarity when compared reflexively to themselves. It also demands that only identical objects have a distance value of 0. If the similarity of two objects is evaluated based on their features and $S$ is a feature representation space, a weaker form (**semi-definiteness**) is given
on the level of the objects since two distinct objects might exhibit the same
feature representation. In that case it holds that if two objects are identical, the
distance between their feature representations is 0. The inverse statement is not
necessarily true.

The symmetry property implies that similarity is a non-directed concept and
for any two objects \( s \) and \( t \), \( s \) is as similar to \( t \) as \( t \) is to \( s \). Many distance
functions employed for similarity search are inherently symmetric even though
symmetry is not necessarily a property of similarity from a perceptual point
of view. Examples of asymmetry in the perception of similarity can be found in
psychological literature. [Tve77] notes that “we say ‘the portrait resembles the
person’ rather than ‘the person resembles the portrait.’ [...] We say ‘an ellipse
is like a circle,’ not ‘a circle is like an ellipse,’ and we say ‘North Korea is like
Red China’ rather than ‘Red China is like North Korea.’” In [TG82] it is observed
that “a prominent object or a prototype is less similar to a non-prominent object
or a variant than vice versa.” Thus, symmetry for a distance function is often a
postulate stemming from advantages in similarity query processing (efficiency)
and not from the notion of similarity that the distance function reflects (effec-
tiveness).

Lastly, the triangle inequality stipulates that there is no shortcut via some
intermediate object \( u \). It is the pivotal property for metric indexing methods
that determine parts of the database as too dissimilar to an object \( s \) by looking
at (possibly precomputed) values of \( d(s, u) \) and \( d(u, t) \) for so-called vantage
or routing objects \( u \). Like the symmetry property, the benefit of the triangle
inequality from an effectiveness point of view can be questioned. [Tve77] gives
an example showing that the triangle inequality is not intrinsically given for
all notions of similarity: “Jamaica is similar to Cuba (because of geographical
proximity); Cuba is similar to Russia (because of their political affinity); but
Jamaica and Russia are not similar at all.” He goes on to note that “although
such examples do not necessarily refute the triangle inequality, they indicate
that it should not be accepted as a cornerstone of similarity models.”

Many commonly employed distance measures (such as the ones reviewed
in Section 1.2.2) are metric distance measures. The transformation-based dis-
tance measures discussed in parts II through V can be metric distance measures
but can also be flexibly adapted such that they reflect non-metric notions of similarity.

### 1.2.2 Examples of Distance Measures

Throughout this work, a number of simple distance measures induced by $L_p$ norms are used in differing contexts. They are also commonly used for distance-based similarity search in the database literature.

**Definition 1.6 ($L_p$ Distance Measures)**

Given an $n$-dimensional vector space $S$ and a parameter $p \in \mathbb{R}$ with $p > 0$, the $L_p$ distance $d_{L_p}$ is defined by

$$d_{L_p}(s, t) = \left( \sum_{i=1}^{n} |s[i] - t[i]|^p \right)^{\frac{1}{p}} \quad \forall s, t \in S.$$ 

Prominent examples of this family of distance functions are $d_{L_1}$, which is known as the Manhattan distance, and $d_{L_2}$, which is known as the Euclidean distance (and what is considered as the crow flies in 3-dimensional spatial space). The $L_p$ distance functions are also defined for $0 < p < 1$ but in that case they are not based on a norm and the triangle inequality does not hold. In addition, the maximum distance $d_{L_\infty}$ is defined as $d_{L_\infty}(r, s) = \max_{i \in \{1, \ldots, n\}} |s[i] - t[i]|$ and measures the greatest difference among all dimensions.
It is apparent from definition of the $L_p$ distances that each dimension of a vector $s$ is only compared with the same dimension of the vector $t$. In distance-based similarity search, such *bin-by-bin* distances can be problematic. Here, the value assigned to a dimension reflects the frequency of a certain (range of) feature(s) observed for an object. For example, a color histogram might encode in its first dimension the portion of pixels in an image that fall in the category light blue while other dimensions might stand for dark blue and magenta. The dimensions are obviously not all perceptually orthogonal or independent, as light blue and dark blue are perceptually similar while light blue and magenta are not. In the example of Figure 1.3, comparing an all light blue image with an all dark blue image yields the same distance as comparing an all light blue image with an all magenta image since bin-by-bin distances ignore the semantics of the dimensions that they compare.

The transformation-based distances that are the focus of this work allow for abandoning this assumption of independence by incorporating a cost measure for transforming one feature into another. In that manner, transforming light blue features into dark blue features can be set to be cheaper than transforming them into magenta features.

The idea of breaking the assumption of independence is also present in a generalization of $d_{L_2}$ called quadratic form distance [HSE+95]. Its efficiency [SK97, ABKS98, BUS10] and effectiveness [ISF98, WBS08, BUS09] has been the focus of past and present database and information retrieval research.

**Definition 1.7 (Quadratic Form Distance Measure)**

Given an $n$-dimensional vector space $S$ and a similarity matrix $A \in \mathbb{R}^{n \times n}$, the quadratic form distance $d_{QFA}$ is defined by

$$d_{QFA}(s, t) = \sqrt{(s - t)^T A (s - t)} \quad \forall s, t \in S.$$ 

Entry $A[i, j]$ encodes the similarity of bin/dimension $i$ and $j$.

While the focus of this work is on transformation-based distances that minimize a cost measure in an intuitive manner, quadratic form distances will surface as approximations and as heuristic initializations in chapters 7 and 11.
Figure 1.4 is a graphical representation of points that are at equal distance from a given central point for the aforementioned distance measures. The ellipsoid iso-distance shapes that quadratic form distances can take are a generalization of the circle-shaped iso-distance surface of $d_{L_2}$. The rotation and proportion of the ellipsoid depends on matrix $A$ with a unit matrix leading to $d_{L_2}$. 
Chapter 2

Distance-Based Similarity Queries

Given a query object \( q \), the task of finding objects in a database \( DB \) that are similar to \( q \) according to some distance measure \( d \) can be formalized in a number of ways. Each formalization characterizes a type of query in the \textit{query-by-example} similarity search framework that reflects a certain use case. Query types vary in their input parameters and in their desired outcome, which is defined in terms of the subset of the database that they retrieve. Common query types include range queries, (k-)nearest-neighbor queries, and ranking queries. The next sections review these query types. A simple sequential algorithm that computes the result set is given for each type. More sophisticated and efficient algorithms exist in the database literature. Prominent approaches such as [HS95, RKV95, KSF+96, SK98] are based on multi-step query processing, dimensionality reduction, data space or distance indexing, or on a combination thereof. These concepts are reviewed in Chapter 3 with a detailed account being available for instance in [Sam05].

2.1 Range Queries

A range query determines all objects in a database that are at least as similar to \( q \) as stipulated by a predefined cutoff or threshold value. In the distance-based similarity model, said threshold defines the maximum distance \( \epsilon \) from \( q \) that objects in the result set may exhibit.
Definition 2.1 (Range Query)
Given a query object \( q \), a range threshold \( \epsilon \), and a database \( DB \), the result of a range query according to distance measure \( d \) is defined as

\[
\text{RangeQ}_{q, \epsilon, d, DB} = \{ o \in DB \mid d(q, o) \leq \epsilon \}.
\]

In the two-dimensional example of Figure 2.1 where \( d \) is the Euclidean distance, three of the ten objects in the database are within the \( \epsilon \)-range indicated by the dotted circle centered around the query object \( q \). Algorithm 2.1 determines \( \text{RangeQ}_{q, \epsilon, d, DB} \) by sequentially checking the distance constraint for every element in the database and including in the result set those elements from the database that are close enough to \( q \).

Threshold-based range queries seem to be very intuitive at first but in practice it is often not easy for a user to determine a suitable threshold \( \epsilon \). A low

\begin{algorithm}
\textbf{Algorithm 2.1: } rangeQ(q, \epsilon, d, DB)
\begin{algorithmic}[1]
\State ResultSet = \emptyset;
\For {o \in DB}
\If {d(q, o) \leq \epsilon}
\State ResultSet = ResultSet \cup \{o\};
\EndIf
\EndFor
\State \textbf{return} ResultSet;
\end{algorithmic}
\end{algorithm}
2.2. Nearest Neighbor Queries

Instead of necessitating a threshold $\varepsilon$, a nearest-neighbor query (NN query) simply returns the object with the highest similarity. If more objects are required, a $k$-nearest-neighbor query ($k$NN query) returns the $k$ most similar objects.

**Definition 2.2 (k-Nearest-Neighbor Query)**

Given a query object $q$, a parameter $k \in \mathbb{N}_1$, and a database $DB$, the result of a $k$-nearest-neighbor query according to distance measure $d$ is defined as

$$NNQ_{q,k,d,DB} = \{ o \in DB \mid \exists S \subseteq DB : |S| = k \land \forall p \in S : d(q,p) < d(q,o) \}.$$

The cardinality of the result set can directly be specified via parameter $k$. Only when multiple objects share the same distance and qualify as the $k^{th}$ neighbor to $q$, the result set is larger than indicated by $k$. In that case, the result set may be forced to include exactly $k$ objects by, for instance, using a tie-breaker.
Algorithm 2.2: nnQ(q, k, d, DB)

for $i = 1$ to $\min(k, |DB|)$ do
  heap.insert( (d(q, DB[i]), DB[i]) ); // sift down
endfor
(kdist, kobject) = heap.front(); // candidate with greatest distance
for $i = k+1$ to $|DB|$ do
  if $d(q, DB[i]) < kdist$ then
    heap.pop(); // remove old $k^{th}$ candidate
    heap.insert( (d(q, DB[i]), DB[i]) ); // sift down new candidate
    (kdist, kobject) = heap.front();
  endif
endfor
ResultSet = $\emptyset$;
while $(dist, object) = heap.pop()$ do
  ResultSet = $\{object\} \cup ResultSet$;
endwhile
return ResultSet;

attribute with a total order. In the heap-based implementation example of Algorithm 2.2, objects with a lower ID are implicitly preferred over those with a higher ID. The heap maintains a set of $k$ nearest neighbors candidates while scanning the database. Objects with a lower distance to $q$ than the current $k^{th}$ nearest neighbor candidate $kobject$ are inserted into the heap according to their distance – displacing $kobject$. Figure 2.2 shows the result of a 3NN query.

2.3 Ranking Queries

In some scenarios, a $k$-nearest-neighbor query is not sufficient as it may not \emph{a priori} be known how many results have to be retrieved. In that case, ranking queries allow for an incremental way of retrieving data from a database. Prominent examples of ranking queries in use are internet search engines, where the results are sorted according to relevance to the query string and grouped into pages of a fixed number of results. If a user is not satisfied after looking through the first page, more results can be requested by going to the next page.
2.3. Ranking Queries

Definition 2.3 (Ranking Query)

Given a query object \( q \), cardinality and iteration parameters \( k \in \mathbb{N}_1, i \in \mathbb{N}_1 \), and a database \( DB \), the result of a ranking query according to distance measure \( d \) is defined as

\[
\text{RankingQ}_{q,i,k,d, DB} = \text{NNQ}_{q,k,d, DB_i}
\]

with

\[
DB_i = DB - \bigcup_{j=1}^{i-1} \text{RankingQ}_{q,j,k,d, DB}.
\]

The first iteration of a ranking query is identical to a kNN query due to \( DB_1 = DB \). For higher iterations, the ranking query is formulated in terms of a k-nearest-neighbor query evaluated on a database from which objects that have been returned in earlier iterations have been removed. Analogously, the stateless Algorithm 2.3 retrieves the \( i \times k \) nearest neighbors but discards the first \((i-1) \times k\) objects of the intermediate result. This simplification is possible if kNN queries return exactly \( k \) objects as in the \( nnQ \) algorithm. More sophisticated stateful algorithms reuse intermediate results from earlier iterations. Figure 2.3 shows the result of the second ranking iteration \( \text{RankingQ}_{q,2,3,d, DB} \) where three more results over those from the 3NN query in Figure 2.2 are retrieved.

Algorithm 2.3: rankingQ(q, i, k, d, DB)

1. return \( \text{nnQ}(q, i \times k, d, DB)[1+(i-1) \times k : i \times k] \);
Chapter 3

Efficient Similarity Query Processing

For large databases and computationally expensive distance measures, the simple sequential algorithms from Chapter 2 often cannot meet the efficiency requirements of real world applications. This chapter gives a quick introduction to concepts from the database research field that are employed to lower computational costs and speed up the search process. A concrete example of how these concepts can be employed and combined for $d_{l_2}$ is given in Chapter 4. Novel techniques specific to transformation-based distances are described in Part III and in Chapter 13.

3.1 Lower Bounds and Multi-Step Processing

The query paradigms from Chapter 2 all exclude from their result set objects from a certain distance on. For range queries, this pruning distance is explicitly given. For nearest neighbor and ranking queries, the distance is incrementally determined as the distance to the object with the largest distance within the intermediate result set during the search. By incorporating a function $d'$ that is a lower bound of the distance function $d$, multi-step query processing approaches filter the database such that fewer objects that are ultimately not part of the result set have to be compared with $q$ using $d$. 
Definition 3.1 (Lower Bound)

Given a distance function \( d : S \times S \rightarrow \mathbb{R}^+ \), a function \( d' : S \times S \rightarrow \mathbb{R} \) is called a lower bound of \( d \) iff \( d'(s, t) \leq d(s, t) \) for all \( s, t \in S \).

If for a query object \( q \) and a database object \( o \) the value given by \( d'(q, o) \) is not below the pruning distance, then \( o \) is not part of the result set of the query as \( d(q, o) \) is always equal to or larger than \( d'(q, o) \). The database can be filtered using \( d' \) in a filter step and the final result set is obtained using \( d \) on the filtered database in what is referred to as a refinement step. The general process is visualized in Figure 3.1 in the form of a Sankey diagram.

If using a lower bound \( d' \) in the query processing is to improve the efficiency of the whole process, \( d' \) has to satisfy certain criteria termed the ICES-criteria in [AWS06]: Indexability, Completeness, Efficiency, and Selectivity. Selectivity pertains to the number of objects that can be excluded from further investigation. If \( d' \) gives values close to \( d \), many objects can be pruned and the filter has a good selectivity. Efficiency regards the computational cost of determining values of \( d' \) which should be lower than the computational cost of determining values of \( d \). Completeness of the filter step requires that no false dismissals occur. If \( d' \) is a lower bound of \( d \), completeness can be guaranteed for commonly used multi-step algorithms. Indexability refers to the desirable ability to support the query processing using an indexing scheme for \( d' \) (cf. Section 3.2).

A number of multi-step query processing algorithms that use a lower bound \( d' \) for a distance function \( d \) exist in the literature. A range query can be an-
answered by evaluating \( \text{rangeQ}(q, \epsilon, d, \text{rangeQ}(q, \epsilon, d', DB)) \) as proposed for example by [AFS93]. That is, first the subset of objects \( o \) in the database that satisfy \( d'(q, o) < \epsilon \) is determined and only for those objects \( d(q, o) < \epsilon \) is checked. [KSF+96] gives a multi-step k-nearest-neighbor algorithm that is based on a lower bound in the filter step. The first \( k \) nearest neighbors according to the filter distance \( d' \) are retrieved and refined with \( d \). Afterward, a range query with the maximum of the \( k \) refined distances is issued using again \( d' \). Finally, the candidates returned by the range query are refined with \( d \) and the \( k \) closest objects are returned as the \( k \) nearest neighbors according to \( d \). In [SK98], a multi-step k-nearest-neighbor algorithm that is optimal in terms of the number of refinement distance computations is given. It is based on a ranking query for the lower bounding distance \( d' \) and iteratively refines candidates until the distance of a candidate according to \( d' \) is greater than the refined distance of the current \( k^{th} \) nearest neighbor candidate according to \( d \).

Depending on the distance function \( d \), finding a suitable lower bound \( d' \) that fulfills the ICES-criteria to a satisfactory degree can be a challenge. A prominent strategy is dimensionality reduction where objects from a high-dimensional feature representation space are mapped to objects in a representation space with a reduced dimensionality. For example, in the case of \( d_{L_2} \) on fixed-binning histograms, dropping any number of dimensions results in a lower-dimensional representation of the histogram on which \( d_{L_2} \) itself is a lower bound. Dimensionality reduction will be the focus of Chapter 4 and Chapter 8.

### 3.2 Indexing Structures

The foremost aim of indexing structures is identical to that of the query processing algorithms based on lower bounds from the last section: to enable a fast identification of a small superset of the final result set. Unlike the lower bounds described in the last section that are based on properties of the distance functions they approximate, indexing structures are designed to achieve this goal by collecting information on the objects contained in the database. Using this data-dependent information, similarity search algorithms that make use of such
indexing structures can rule out or prune parts of the database without having to look at all of the database objects.

The number of indexing structures that have been proposed for supporting distance-based queries has grown considerably with every year over the past few decades. They differ in many ways, including the nature of the data they index, the distance measures and query types they support, the query loads they are optimized for, and in their support for object insertions and deletions. A recent account is given in [Sam05]. As even that immensely extensive work cannot cover all indexing structures for distance-based query processing, no such attempt shall be made here. Instead, a quick overview of those concepts and indexing structures important for the following parts is given in this section. Further details will be provided in the according chapters as required.

From a database research perspective, the R-Tree [Gut84] and its variants are typically considered the most well-known secondary memory indexing structures for multi-dimensional, spatial data (i.e., points or objects with an extent in a vector space equipped with a distance function). The general idea of R-Trees and many other hierarchical indexing structures is to map spatial proximity (cf. Figure 3.2(a)) of the objects in the database to topological proximity (cf. Figure 3.2(b)) in the data structure by grouping objects according to their membership in nested bounding geometries. In the case of the R-Tree, an inner node of this multi-way tree describes each of its subtrees in terms of an axis-parallel minimum bounding rectangle (MBR) and an identifier that allows for the retrieval of a subtree from the storage system. The leaf nodes contain
3.2. Indexing Structures

**Algorithm 3.1: rtree-rangeQ(q, ε, d, minDist, RTree)**

```plaintext
return rec-rtree-rangeQ(q, ε, d, minDist, RTree.rootnode);
```

**Algorithm 3.2: rec-rtree-rangeQ(q, ε, d, minDist, node)**

```plaintext
ResultSet = ∅;
if node.isleaf then
  for o ∈ node.dataobjects do
    if d(q, o) ≤ ε then
      ResultSet = ResultSet ∪ {o};
    endif
  endfor
else
  for childnode ∈ node.childnodes do
    if minDist(q, childnode.mbr) ≤ ε then
      ResultSet = ResultSet ∪ rec-rtree-rangeQ(q, ε, d, minDist, childnode);
    endif
  endfor
endif
return ResultSet;
```

the database objects (vectors or extended objects and potentially further information). The MBRs of R-Trees can overlap and the capacity of each node is restricted as it is to be stored in secondary memory where the capacity is chosen as a multiple of the smallest addressable unit of memory of the storage system. The R-Tree family is large [SRF87, BKSS90, BKK96], with new members such as [KCK01, BS09] joining even a quarter of a century after Guttman’s publication.

When processing a similarity search query, the MBRs are used to decide whether objects in the subtree can be pruned from the search. For this purpose, a smallest possible distance from the query to the bounding rectangle of a subtree is defined. This so-called MinDist is a lower bound for all objects in said subtree and can thus be compared with the pruning distance of a search. The definition of the MinDist depends on the distance measure that is employed. Algorithms 3.1 and 3.2 list pseudo-code for simple recursive range query processing akin to the rectangular overlap query from [Gut84]. Similarly, nearest
neighbor algorithms such as the Depth-first Traversal algorithm of [RKV95] and the Priority Search algorithm described in [HS95] use the R-Tree (and other indexing structures) to prune whole branches of the tree once the feature space region they represent can be guaranteed to not include the nearest neighbor.

A second group of indexing structures often employed to speed up the similarity search process relies on the metric properties – in particular on the triangle inequality property – that many distance functions possess. Instead of grouping objects with explicit bounding geometries, metric indexing structures like the M-Tree [CPZ97] gather information regarding the distance of objects in the database to a set of (sometimes hierarchically structured) pivot or vantage points. Given the distance from the query to a pivot point and the precomputed distance from the pivot point to a database point, the distance required for the similarity search (i.e., from the query to the database object) can be estimated using Definition 1.5. By storing the maximum distance from a pivot point to all data objects in according subtrees, an implicit sphere around the pivot point is established that allows for pruning of the search space. As metric indexing structures only require the distance function to be metric, they are not restricted to the vector space model.

When trying to index higher-dimensional points (where “higher” starts at about 5–10 for many indexing structures), the pruning capability of hierarchical indexing structures rapidly declines due to the curse of dimensionality. Both the R-Tree and the M-Tree suffer from a large degree of overlap of the MBRs or implicit spheres. The VA-file [WSB98] exemplifies efforts to speed up the process of scanning the whole database while searching the database. Here, dimension-wise quantization is used to approximate the location of objects as within a certain rectangular region. As the quantized information is compact and as the dimension-wise boundaries of the rectangular regions are known in advance, a scan over all quantized data is comparatively fast and partial distance information can be precomputed. Only when the approximate location of an object is not enough to establish whether it is part of the result set must its full representation be retrieved and compared with the query object.
Chapter 4

An Example for Similarity Search Using Simple Distance Measures

This chapter gives an example of how various techniques introduced in Part I can be put together to forge a similarity query system (cf. [BWS06]) that is tailored to support efficient nearest neighbor query processing based on the Euclidean distance measure. In a multi-step query processing framework, Voronoi cells – which reflect local neighborhood information – are indexed to create a fast filter for the nearest neighbor search.

4.1 Introduction

Utilizing spatial index structures on secondary memory for nearest neighbor search in high-dimensional spaces has been the subject of much research. However, a growing number of applications demanding a high query throughput stand to benefit from a shift toward index structures tailored for main memory indexing. Server-based similarity search services are examples of such types of applications. In this scenario a large number of querying clients demands a low response time from a server that bases its service on static or semi-static data. Other examples include algorithms that implement a classification or density-based clustering method based on the concept of nearest neighbors. The “(re)index rarely, query frequently” behavior of those applications allows
for an extensive preprocessing phase in which the index is built from scratch. The volatile nature of main memory is not a disadvantage in this scenario as the data is stored on secondary memory and only a copy is loaded into main memory to be accessed at a high frequency. The economic impact of reserving a few hundred megabytes or a few gigabytes of RAM to host larger indexes in main memory is diminishing fast while the increase in efficiency that goes along with it can be significant.

One approach that relies on an extensive preprocessing step consists of indexing the solution space for nearest neighbor queries in the form of approximate Voronoi cells \([\text{BEK}^*98, \text{BKKS00}]\). Voronoi cells describe a covering of the underlying feature representation space such that each data object is assigned to the cell that contains all its possible nearest neighbor locations. A preprocessing step is used to approximate the complex Voronoi cells with simpler high-dimensional axis-aligned bounding rectangles (i.e., rectangular cuboids) in order to enable low query response times. While its low CPU-utilization makes the approach a natural candidate for main memory indexing, it resists attempts to incorporate effective dimensionality reduction techniques. Straightforward solutions prove to be unsuitable. This severely limits the application domain of the approximate Voronoi approach as approximate Voronoi cells in high-dimensional spaces are neither feasible to compute nor efficient for indexing.

Sections 4.1.1 and 4.2 review related work and the approximate Voronoi ap-
4.1. Introduction

The problems of incorporating dimensionality reduction techniques to support multimedia similarity search are described in Section 4.3.1 and approximation-based methods to efficiently overcome these difficulties are introduced. A second dimensionality reduction (on the level of the cuboids) presented in Section 4.3.2 improves response times through limiting the dimensionality of the bounding cuboids themselves. The cuboids in the reduced dimensionality are indexed by facilitating either hierarchical or bitmap-based index structures in main memory as described in Section 4.3.3. It is possible to find a complete set of nearest neighbor candidates for a query point in a filtering step through simple point-in-cuboid tests (cf. Figure 4.1). The significant performance improvements over other approaches achieved for the Voronoi-based technique through these two dimensionality reduction steps are shown in Section 4.4 for real world datasets.

The result is a filter-and-refine query processing system for rapid nearest neighbor search based on the Euclidean distance with speedup factors of up to five versus other evaluated RAM-resident indexing structures.

4.1.1 Related Work

The concept of partitioning a space to describe nearest neighborhood information utilized in this chapter was developed in the early twentieth century by G. Voronoi [Vor08] and is a generalization of the 2- and 3-dimensional diagrams already used by Dirichlet in [Dir50] and informally described by Descartes in [Des44]. It is still a widespread and important topic of extensive research in the field of computational geometry [AK00]. A first algorithm that efficiently computes the nearest neighbor of a query based on a Voronoi diagram was proposed in [DL76]. The algorithm performs nearest neighbor queries on \( m \) points in the 2-dimensional Euclidean plane in worst-case-optimal time complexity of \( O(\log m) \). However, the algorithm does not extend well to dimensionalities higher than two. In [BEK+98, BKKS00], a technique using approximate Voronoi cells was introduced and enabled the use of datasets with low to medium dimensionalities. The technique also reaches its limits with increasing dimensionality. By enabling dimensionality reduction to be used with the ap-
proximate Voronoi approach, the techniques in this chapter expand on the work in [BEK+98, BKKS00].

Due to main memory becoming significantly larger in capacity and significantly cheaper in price, research interest on indexes in main memory has been renewed in the database community. Well-known secondary memory indexing structures that utilize the concepts reviewed in Section 3.2 have been investigated and modified for the main memory scenario. For example, the CR-Tree [KCK01] improves on the cache-consciousness of the R-Tree [Gut84] by applying MBR compression techniques while both the pkT-Tree [BMR01] and the CSB+-Tree [RR00] focus on low-dimensional main memory indexing.

4.2 Preliminaries

4.2.1 Voronoi Cells

A Voronoi diagram for a set of \( m \) points in a given space is a covering of said space by \( m \) cells that indicate the nearest neighbor areas of the points. It is thus directly tied to the problem of finding nearest neighbors. Figure 4.2(a) shows such a partitioning for six points in the Euclidean plane. For each point, the respective surrounding cell describes the area for which that point is closer than any of the other points. Given a query position \( q \) in the plane, the nearest neighbor can be found by determining the cell that includes that position. As

Figure 4.2: A Voronoi diagram and the definition of a Voronoi cell
4.2. Preliminaries

Algorithm 4.1: voronoi-nnQ(q, VD^{DB,S})

\begin{algorithmic}
\State \textbf{for} VC^{p,S} \in VD^{DB,S} \textbf{do}
\State \hspace{1em} \textbf{if} q \in VC^{p,S} \textbf{then}
\State \hspace{2em} \textbf{return} \{p\};
\State \hspace{1em} \textbf{endif}
\State \textbf{endfor}
\end{algorithmic}

long as q remains inside the same cell, its nearest neighbor does not change. The edges and vertices in the diagram describe positions for which more than one point is at minimum distance. Voronoi cells can formally be described using the concept of half spaces as follows.

\textbf{Definition 4.1 (Voronoi Cell)}

Given a space S, a metric distance function \( d : S \times S \rightarrow \mathbb{R}^+ \), and a finite set of points \( DB \subseteq S \), a Voronoi cell \( VC^{p,S} \) for point \( p \in DB \) is defined as the intersection of S and \( |DB| - 1 \) half spaces:

\[
VC^{p,S} = S \cap \left( \bigcap_{r \in (DB-\{p\})} HS^{p|r} \right)
\]

where

\[
HS^{p|r} = \{s \in S|d(s, p) \leq d(s, r)\}.
\]

\textbf{Definition 4.2 (Voronoi Diagram)}

A Voronoi diagram \( VD^{DB,S} \) is defined as the set of the Voronoi cells:

\[
VD^{DB,S} = \{VC^{p,S}|p \in DB\}
\]

Figure 4.2(b) shows the half space intersections for a Voronoi cell in light blue where the half space separation lines are shown as dashed lines. Algorithm 4.1 exemplifies how a Voronoi diagram can be used to find the nearest neighbor of a query.
Algorithm 4.2: approx-voronoi-nnQ(q, d, AVD^DB,S)

1. CandidateSet = ∅;
2. for AVC^p,S ∈ AVD^DB,S do
   3.    if q ∈ AVC^p,S then
   4.        CandidateSet = CandidateSet ∪ {p};
   5.    endif
3. endfor
4. return nnQ(q, 1, d, CandidateSet) ;  // see page 22

4.2.2 Approximate Voronoi cells

In more than two dimensions (as is the case for feature representation spaces in many multimedia similarity search applications) the cells become highly complex [Kle80, Sei87]. Due to this complexity, neither computing nor storing or inclusion-testing is efficient for nearest neighbor search directly based on these cells. Therefore, the preprocessing step in this chapter determines approximate Voronoi cells of lesser complexity without requiring the exact representations of the latter to be known while still allowing for fast nearest neighbor search. To avoid false dismissals during nearest neighbor query processing, each approximate cell must be a superset or bounding geometry of the original Voronoi cell. Algorithm 4.2 shows how an approximate Voronoi diagram AVD (consisting of approximate Voronoi cells AVC) can generally be used to find the nearest neighbor of a query in a filter-and-refine manner. Here, the filter is not based on a lower-bounding distance function of d. Instead, an inclusion test is used to rule out those objects for which query q falls outside their according approximate nearest neighbor region. For the remaining objects in the candidate set, a nearest neighbor query using d is performed.

Among the potential bounding geometries, axis-aligned bounding cuboids offer several advantages. For n-dimensional points, they enable inclusion tests in O(n) time, they can be stored in O(n|DB|) space, and they are computable through well-studied linear optimization algorithms [PTVF92].
4.2. Preliminaries

4.2.3 Computation of Approximate Voronoi Cells

A bounding cuboid of a Voronoi cell $V C_{p,S}$ ($p \in \mathbb{R}^n$) can be computed by solving $2n$ linear optimization problems. In linear optimization a linear objective function is maximized or minimized over a range of values restricted by a set of linear constraints. In the context of Voronoi cell approximation, the required linear constraints are defined by

- the set of half spaces outlining the cell and
- the space boundaries of $S$.

For the approximation approach to Voronoi cells chosen here, it is required that $S \subset \mathbb{R}^n$ be of convex shape such as the $n$-dimensional unit hypercube $[0, 1]^n$. All potential query points are also assumed to be from within $S$.

The $n$ objective functions are used to find the outermost points in each of the $n$ dimensions of a cell $V C_{p,S}$ described by the linear constraints. For this purpose, functions $f_1$ to $f_n$ with $f_i(x_1, ..., x_n) = x_i$ are each minimized and maximized once per Voronoi cell. The extremal values directly represent the respective boundaries of the cuboid that tightly bounds the Voronoi cell. The space boundaries must be added to the set of constraints to avoid that these extremal values extend outside the data space in some of the dimensions – potentially causing the cuboid to unnecessarily grow within $S$ for other dimensions.
Only a subset of all potential half spaces may be used in order to significantly speed up the calculation of the bounding cuboids during the preprocessing step. Half spaces that are redundant due to not constraining a Voronoi cell can be left out without affecting the result of the cell approximation. Leaving out non-redundant half spaces leads to a non-minimum bounding cuboid, which potentially introduces more nearest neighbor candidates and slows down nearest neighbor searches by requiring more refinement calculations but never misses a valid solution. Therefore the choice of the subset of half spaces is very important. In [BEK+98] some heuristics for the choice of an appropriate subset of half spaces are introduced. This chapter concentrates on a heuristic that selects a limited number of nearest neighbor points from DB for each data point \( p \in \text{DB} \) and uses the corresponding half spaces to approximate the Voronoi cell \( \text{VC}^p \) since the half spaces defined by the nearest neighbors of \( p \) are likely to actually restrict the cell. For the range of parameters evaluated in Section 4.4, restricting the number of nearest neighbors used for the approximation to 1% of the database has proven to yield good results.

Figure 4.3(a) shows the approximate cell belonging to object \( p \) where all half spaces were used while in Figure 4.3(b) the half space \( \text{HS}^{p\mid p'} \) is left out, resulting in a slightly larger cuboid.

### 4.3 Dimensionality Reduction for the Voronoi Approach to Nearest Neighbor Search

The potentially high dimensionality of the object representation in multimedia applications hinders the efficient utilization of the Voronoi approach to nearest neighbor search in three ways.

First, a high dimensionality results in the data representation space being sparsely populated with data points. Thus the bounding cuboids of the Voronoi cells are relatively large in this case as only comparatively few other cells are available to restrict each cell in all possible directions. In extreme cases, all cuboids overlap the complete space since each cell includes points on both the upper and lower space boundary of each dimension. These unrestricted cuboid
4.3. Dimensionality Reduction for the Voronoi Approach to NN Search

Figure 4.4: Two-step strategy for reducing the dimensionality

dimensions are useless for nearest neighbor searches, since they never cause any point to be dismissed as a nearest neighbor candidate for any query.

Second, computing the cell-bounding cuboids becomes more expensive as each dimension adds a new variable to the linear optimization process and more linear constraints are required to describe the cells of points in $S$ as the total number of facets in $VD_{DB,S}$ increases.

Finally, once the Voronoi cell approximations have been computed, they are to be indexed to efficiently answer queries. For cuboids in a high dimensionality, hierarchical index structures such as the R-Tree or X-Tree are bound to experience a deteriorating efficiency caused by the effects of the curse of dimensionality.

A partial solution to these problems lies in applying dimensionality reduction techniques as a special form of approximation. For the Voronoi approach, this can be performed on two levels as summarized by Figure 4.4.

For the first dimensionality reduction step ($n$ to $n'$), all linear dimensionality reduction techniques (including PCA, Random Projection, Wavelet Transforms, Discrete Fourier Transform, etc.) can be used in the approach described here. Due to its optimality regarding the mean squared error, its predominant position in practical applications, and its two-part output (rotation matrix and variances), which is useful in section 4.3.2, the following sections focus on PCA. In this first reduction step, some of the nearest neighborhood information is given up in order to construct the Voronoi cell-bounding cuboids more efficiently and
lessen the effects of the *curse of dimensionality*. A sizable number of dimensions can often be removed in this way while only introducing minor inaccuracies. The feature extraction process employed for a similarity search application can result in dimensions that are insignificant to the application at hand. This is the case when one dimension is dominated by other dimensions either due to differences in the variance of the dimensions or due to correlation effects. Introducing some minor inaccuracies through this first dimensionality reduction can often be acceptable in return for a more efficient preprocessing step and for faster query processing.

In the second dimensionality reduction step described in section 4.3.2, the dimensionality of the resulting cuboids can be further reduced prior to indexing. Unlike the first reduction step, this reduction does not influence the effectiveness but only the efficiency of the nearest neighbor search process.

The overall dimensionality reduction process outlined by Figure 4.4 is thus to first reduce the data representation dimensionality from $n$ to $n'$ in a lossy fashion by as much as the application allows for and then to reduce the approximation dimensionality from $n'$ to $n^*$ in order to improve the efficiency of the search process while ensuring that the correct answer on the level of the chosen $n'$ is still returned.

Looking at the ICES filter quality criteria (cf. page 26), the filter-and-refine process is complete (C) on level of $n'$ and an inclusion test for a cuboid with reduced dimensionality is efficient (E). The indexability (I) will be handled in Section 4.3.3 and the selectivity (S) is evaluated in the experiments.

### 4.3.1 The Bounding Constraints Problem

Unlike other nearest neighbor algorithms, the Voronoi approach described here depends on the data space being included in a polytope whose facets are used to define the outer constraints for the linear optimizations of the bounding cuboid computation. Dimensionality reduction techniques for the Voronoi approach have to take this property into account.

For similarity search based on fixed-binning histograms examined in this chapter, all histograms $h_{p_i} = (h_{p_i}[1], \ldots, h_{p_i}[n])$ of $p_i \in DB$ share a common sum
4.3. Dimensionality Reduction for the Voronoi Approach to NN Search

For these vectors, there is an \((n - 1)\)-dimensional convex polytope with \(n\) vertices and \(n\) facets that includes all vectors. After rotating and projecting all points to eliminate the redundant dimension \(h^p_i[n] = 1.0 - \sum_{j=1}^{n-1} h^p_i[j]\), the \(n\) vertices of the polytope consist of the accordingly transformed unit vectors. The transformed polytope serves as the data representation space \(S\) that confines the Voronoi cells and contributes linear constraints to the approximate Voronoi cell computation. Figure 4.5 illustrates the transformation for the case of \(n = 3\) where all points are enclosed by the three lines between the three transformed unit vectors in a two-dimensional subspace.

In practical applications, the originally sum-normalized data is often linearly transformed. Scaling of individual dimensions is used to compute weighted distances and both rotations and projections are common in dimensionality reduction (PCA, DFT, Random Projection and others). The aim is to find the transformed convex polytope defining the data space – in particular after projections into a dimensionality \(n' < n\). A linear transform of a convex polytope is another convex polytope where the vertices of the transformed polytope are (a subset of) the transformed original vertices. Thus, one way to find the transformed polytope is to transform all original polytope vertices \(P\) and then find the convex hull for those points \(P'\). This approach has a worst-case time complexity of \(O(m \log m + m \lfloor n'/2 \rfloor)\) for \(m\) points in the (possibly lower) transformed dimensionality \(n'\) [Ede87].

The high complexity of the convex hull leads to another problem. Each facet
Figure 4.6: Number of facets for convex hulls in the projected dimensionality $n'$. The number of facets of the convex hull produces a constraint for the linear optimizations for each Voronoi cell. Hence, that number must be low for practical reasons. Contrary to that, a convex hull with $m$ vertices in $n'$ dimensions can have in the order of $O(m\lfloor n'/2 \rfloor)$ facets [Ede87]. Figure 4.6 shows these values computed via the QHull algorithm [BBDH96] for two real world datasets used in Section 4.4.

While the convex hulls for the phoneme dataset remain sufficiently simple, the image histogram dataset quickly goes beyond values reasonable for a practical computation of the Voronoi cell-bounding cuboids. Due to that fact, a number of methods to conservatively approximate the convex hull are introduced in the following pages. An approximation of the convex hull of a set of points is called conservative in this context if all points are also contained in the approximation of the convex hull.

**A Bounding Cuboid Approximating The Convex Hull**

A simple way to conservatively approximate the convex hull of a set of points $P'$ with $(n' - 1)$-dimensional planes is to find a bounding cuboid for the hull. This can be done by determining the minimum and maximum values among the set $P'$ for each dimension. The resulting $2n'$ planes defined by the cuboid facets are suitable as constraints for the Voronoi cell approximations. However, the shape of the convex hull of $P'$ can be quite different from an axis-aligned cuboid. If a greater precalculation cost is acceptable to improve the selectivity for queries, it is worth finding a more complex but closer approximation of the convex hull.
4.3. Dimensionality Reduction for the Voronoi Approach to NN Search

Figure 4.7: Planes approximating a convex hull

**Tilted Planes Approximating the Convex Hull**

A potentially closer approximation can be found by using the vertices of the bounding cuboid. For each of the $2^n'$ vertices, the adjacent $n'$ vertices span a hyperplane with a normal vector $v$ as depicted in Figure 4.7(a) for vertex $x$ of the cuboid bounding the convex hull. Each such tilted hyperplane is then pushed outwards along its normal vector until all points in $P'$ are located either on the hyperplane or behind it as defined by the orientation of the normal vector. This plane-fitting algorithm has a time complexity of $O(n'|P'|)$.

**Inner Polytopes Approximating the Convex Hull**

Like the bounding cuboid approach, the tilted planes method makes little use of the geometric shape of the convex hull being approximated. The normal vectors of the hyperplanes are only influenced by the total extent of the convex hull in each dimension. A variant is proposed here that attempts to find more suitable normal vectors for the plane fitting. The idea is to define a less complex convex polytope residing inside the convex hull of $P'$ that still reflects its general shape. Once the facets of the inner polytope have been determined, they are pushed outwards along their normal vectors to include all the points in $P'$. 
An Example for Similarity Search Using Simple Distance Measures

The polytope used in this proposal is defined through its vertices, which form a subset of the vertices of the convex hull of \( P' \).

**Definition 4.3 (Set of Extremal Points)**

*Let \( R = \{ r_1, r_2, \ldots \} \) be a finite set of \( n \)-dimensional points. The set of extremal points \( \text{Ext}^R \) is defined as*

\[
\text{Ext}^R = \text{ExtMin}^R \cup \text{ExtMax}^R
\]

*with*

\[
\text{ExtMin}^R = \{ r_i \in R | \exists k \in \{1, \ldots, n\} : \forall r_j \in R - \{ r_i \} : (r_i[k] < r_j[k]) \lor ((r_i[k] = r_j[k]) \land (i < j)) \}
\]

*and*

\[
\text{ExtMax}^R = \{ r_i \in R | \exists k \in \{1, \ldots, n\} : \forall r_j \in R - \{ r_i \} : (r_i[k] > r_j[k]) \lor ((r_i[k] = r_j[k]) \land (i < j)) \}
\]

Intuitively, these are the points that are located on an axis-aligned minimum bounding cuboid of the set \( R \). If more than one point is located on the same facet of the cuboid, a tie-breaker is used. Given \( \text{Ext}^{p'} \), points \( \{ p'_1, \ldots, p'_n \} \subseteq \text{Ext}^{p'} \) are selected for each vertex \( x \) of the bounding cuboid of \( P' \) as illustrated by Figure 4.7(b). These are the points that were included in \( \text{Ext}^{p'} \) due to their position on a facet of the cuboid that has \( x \) as one of its vertices. If none of the points are duplicate for vertex \( x \), the \( n' \) points define a facet of an inner polytope of the convex hull which can then be pushed outwards by the plane-fitting method detailed earlier. In higher dimensionalities with relatively few points in \( P' \), it often happens that one point is extremal for more than one dimension and thus the maximum number of \( 2^{n'} \) facets is rarely reached for the inner polytope. In the illustration of Figure 4.7(b), this would be the case if one of the three points selected for \( x \) was either on an edge of the cuboid or on a vertex of the cuboid. The resulting inner polytope would then have five or four vertices only versus the expected six vertices (three from \( \text{ExtMin}^{p'} \) and \( \text{ExtMax}^{p'} \) each).
4.3. Dimensionality Reduction for the Voronoi Approach to NN Search

Greedy Point Selection

Further inner polytopes can be defined by using other methods to select a subset $T \subset P'$ and then constructing the less complex convex hull for $T$. The facets of the hull are then pushed outwards along their normal vectors until they include all points in $P'$. In order for $T$ to represent the general shape of $P'$, a greedy heuristic for selecting points $\{t_1, ..., t_{|T|}\}$ is proposed: choose any one point in $P'$ that is located on the convex hull of $P'$ as $t_1$. Then choose $t_{i+1}$ as the point $p$ from $(P' - \{t_1, ..., t_i\})$ with the greatest accumulated distance $\sum_{j=1}^i d(p, t_j)$. The greedy selection algorithm runs in $O(n' \cdot |P'| + n' \cdot |T|^2)$ time followed by the computation of the convex hull for $|T|$ points.

Combinations and Variations

The strategies described for finding a convex polytope that conservatively approximates the space in which data points potentially reside offer practical alternatives to the convex hull with its high facet count. More approaches can directly be derived from the previous approximations. An intersection of two conservative, convex approximations yields another conservative, convex approximation. Thus, all approaches can be combined by using constraining planes from more than one approach. Adding the $2n'$ axis-aligned facets of the convex hull-bounding cuboid to constraints from another approximation hardly induces any computational effort but helps reduce the size of the also axis-aligned Voronoi cell-bounding cuboids and ensures that the data space is bounded appropriately in all dimensions. Similarly, the hyperplanes retrieved from a greedy point selection can be augmented by both the cuboid and the inner polytope hyperplanes.

4.3.2 Reduction of the Bounding Cuboid Dimensionality

The dimensionality reduction discussed enables efficient computation of the bounding cuboids by projecting the data points from dimensionality $n$ to a space of dimensionality $n'$. However, it does so by sacrificing nearest neighborhood information. Though justifiable to some extent, it is often not viable to reduce
the dimensionality to a level where indexing the cuboids is efficient. Therefore, a further reduction to a dimensionality \( n^* < n' \) (cf. Figure 4.4) is proposed that does not sacrifice the effectiveness of the approach.

First, a query vector \( q \) is transformed and projected to \( q' \) in dimensionality \( n' \) by the same method used to reduce the data dimensionality. Without the cuboid projection, vector \( q' \) would then be tested for inclusion in the Voronoi cell-bounding cuboids defined by their respective upper and lower boundary vertices. Instead, after computing the Voronoi cell-bounding cuboids for the projected data points in dimensionality \( n' \), those cuboids themselves are projected to dimensionality \( n^* \) by dropping some of the dimensions. This produces the same nearest neighbor as in the case of the cuboids in dimensionality \( n' \) as dropping dimensions relaxes the query inclusion test in Algorithm 4.2. While no candidates are lost in this process, there might however be additional false positives. Formally, the characteristic function of the filter regarding all \( n' \) dimensions is a lower bound of the characteristic function of the filter regarding only a subset of \( n^* \) dimensions.

**Definition 4.4 (Characteristic Function of a Filter)**

The characteristic function \( \chi : S \times S \to \{0, 1\} \) of a filter for objects from \( S \) returns 0 on \( (q, o) \) if object \( o \) is filtered out with respect to \( q \) and 1 if it passes the filter.

**Definition 4.5 (Characteristic Function for the approximate Voronoi filter)**

Let \( DB \subset S \) be a finite set of \( n \)-dimensional points within space \( S \), \( E \subseteq \{1, ..., n\} \) a set of dimensions, and \( AVD^{DB,S} = \{AVC^{o,S} | o \in DB\} \) the set of Voronoi cell-bounding cuboids with lower and upper boundary vertices \( l^o \) and \( u^o \). Then the characteristic function \( \chi^E : S \times S \to \{0, 1\} \) for the approximate Voronoi inclusion-test filter based on dimensions \( E \) is defined as

\[
\chi^E(q, o) = \begin{cases} 
1 & \text{if } (o \in DB) \land \forall e \in E : l^o[e] \leq q[e] \leq u^o[e] \\
0 & \text{otherwise}
\end{cases}
\]

If \( E^* \) is a subset of dimensions \( E' \) then \( \chi^{E^*} \) is an upper bound of \( \chi^{E'} \) since a filter based on \( E^* \) instead of \( E' \) might allow more but not fewer objects to pass. The selectivity of the filter step (cf. Figure 4.1) worsens as more dimensions are
dropped. However, fewer dimensions can result in a faster filter step execution. The following question then arises: Which dimensions are to be dropped and which ones are to be retained? Unfortunately, there are \( \sum_{e=1}^{n'} \binom{n'}{e} \) combinations to consider in an exhaustive search. The search space must be vastly reduced by concentrating on promising combinations.

**Empirical Selectivities for Finding Dimensions to Retain**

Given a sample query workload \( Q \), average selectivity values can be used to find dimensions with a good selectivity. The aim is to find a filter for which both the number of dimensions and the average selectivity is low (i.e., the filter shall only let a low number of candidates pass).

**Definition 4.6 (Empirical Selectivity)**

Given a database \( DB \) and a query workload \( Q \), the empirical selectivity of a filter is defined via its characteristic function \( \chi \) as

\[
\frac{1}{|Q|} \sum_{q \in Q} \frac{1}{|DB|} \sum_{o \in DB} \chi(q,o).
\]

The heuristic proposed in this section vastly reduces the search space by assessing the selectivity of each of the \( n' \) dimensions without regard to the other dimensions. A high empirical selectivity value for \( \chi^{(k)} \) with \( 1 \leq k \leq n' \) indicates that the \( k^{th} \) dimension is unable to filter out many objects. Using a greedy selection of dimensions in the order of the worsening individual empirical selectivity is optimal if the filtering events are mutually independent. In that case, the empirical selectivity of two combined dimensions is expected to be close to the product of both individual empirical selectivities. While statistic independence cannot be expected in general, the PCA utilized for the reduction from \( n \) to \( n' \) dimensions does eliminate the covariance between the \( n' \) dimensions (which in the case of a multivariate normal distribution entails independence). After PCA reduction to a dimensionality of 10 for the 27-dimensional image histogram dataset from the experimental section, the two dimensions with the highest variance exhibit an empirical selectivity of 15% and 10.8%. The expected combined selectivity would be close to \( 15\% \cdot 10.8\% = 1.62\% \) in case of
independence. The measured value was 1.95%. With an increasing number of dimensions, the relative gap widens to an expected 0.0027% for 10 dimensions and 0.0596% measured as additional dependencies are introduced. Still, the proposed order of dropping dimensions resulted in combined selectivities rarely outperformed by spot samples of the same number of dimensions, which justifies using the heuristic proposed in this section.

Using Variances to Find Dimensions to Retain

To avoid having to find a suitable query set $Q$ and compute empirical selectivities, another simplification can be employed when PCA has been used to reduce the data dimensionality. As a by-product of the PCA, the variances for each dimension of the rotated and projected data points are known. Instead of using the worsening selectivity order to pick dimensions to retain, the descending order of the variances can be used. While a high variance for a dimension does not necessarily imply a good selectivity for that dimension, it can be a good indicator as dimensions with a low variance tend to produce Voronoi cells that stretch far in said dimensions. Measured correlation coefficients for inverted variances and measured selectivities of individual dimensions were 0.957, 0.504 and 0.937 for the three real world datasets used in the experiments.

On the Number of Dimensions to Retain

The question of how many dimensions are to be retained remains. The decision depends on the index structure used. One possibility would be to produce up to $n'$ indexes using 1 to $n'$ dimensions selected via the methods proposed above and then select the one index with the best average query performance. Without further information about the index structure, this might be the only method to pursue. Experiments in Section 4.4 show that the X-Tree greatly benefits from the additional projection while the evaluated bitmap-based structure does not.
4.3. Dimensionality Reduction for the Voronoi Approach to NN Search

4.3.3 Main Memory Indexing for the Voronoi Approach

Data page access and read times are not the foremost cost factors of performing nearest neighbor queries when the index structure is stored in main memory. The querying time can be decomposed into a filter part (decide which points in $DB$ need further examination) and a refinement part (compute the distances to the query point $q$). Both parts vary widely for the available algorithms and index structures. While the linear scan (cf. page 22) does not spend any time on filtering but computes the distances from $q$ to all data points in $DB$, the Priority Search algorithm [HS95] for hierarchical index structures carefully considers which data points warrant a distance computation.

The filter step of the Voronoi approach presented here is based on point-in-cuboid tests. Since the Voronoi cell-bounding cuboids are axis-aligned, the R-Tree family (normally based on secondary memory) is an obvious candidate to support the filtering step. Due to its better suitability for data with a dimensionality beyond approximately six, the X-Tree [BKK96] was chosen to be adapted to run in main memory for tests in Section 4.4. In addition, a bitmap-based index structure described in [GPB03] was implemented. The authors originally used the Bitvector Index for fixed-range queries in a memory-resident similarity search system but its design makes it a promising option for Voronoi-based nearest neighbor queries.

For each dimension $i$ of the space to be indexed, the Bitvector Index stores a number $b$ of bitvectors $BV_{i,1}$, ..., $BV_{i,b}$ together with $b + 1$ delimitation values $b_{i,0}, b_{i,1}, ..., b_{i,b}$. When indexing $m$ cuboids, each $BV_{i,j}$ has $m$ bit entries denoting which of the cuboids cover part of the interval $[b_{i,j-1}, b_{i,j}]$ in dimension $i$. In the example depicted in Figure 4.8, only the cuboids $c_2$, $c_3$ and $c_4$ overlap the hatched interval belonging to $BV_{1,2}$.

When considering the query point $q$ in Figure 4.8, only cuboids that overlap the crossed area are of interest during a nearest neighbor query using Voronoi cell-approximating cuboids. These can be found by using a bit-wise ‘and’ operation on $BV_{1,2}$ and $BV_{2,1}$. Hence, only $c_2$ must be further examined in the example. A nearest neighbor algorithm using this index structure and Voronoi cell-approximating cuboids goes through four steps.
1. Find interval in which the query point falls (binary search per dimension).

2. Combine the respective bitvectors via bitwise ‘and’.

3. Retrieve list of indices represented by the set bits in resulting bitvector.

4. Find nearest neighbor among data points belonging to the list of indices.

On $m$ data points in $n'$ dimensions with cuboids of dimensionality $n^*$, the time complexity for the first three steps (the filter) is $O(ld(b) \cdot n^* + n^* \cdot m)$ and the fourth step has a worst-case time complexity of $O(n' \cdot m)$. The space complexity of the index is $O(n^* \cdot b \cdot m)$. While the worst-case time complexity is linear in $m$ for fixed $n'$, $n^*$, and $b$, the index has the potential to achieve a significant performance boost compared with the sequential scan through packing for example 32 bits of a bitvector into an unsigned integer of that length. Thus, instead of computing $32 n'$-dimensional floating point distances only $n^* - 1$ 32-bit integer ‘and’ operations are required to create an integer that has its bits set to 1 for a superset of cuboids that contain the query point $q$. Only for these cuboids
4.4 Experiments

To test the different proposed methods of producing outer constraints for the data space from Section 4.3.1, various empirical selectivities were computed using a query set $Q$ consisting of 7,510 out of 207,510 image histograms projected from $n = 27$ to the first 10 principal components. This left 200,000 histograms for the data point set $DB$.

Figure 4.9 shows that the convex hull produced a total of 9,263 constraints. Using a higher dimensionality $n'$ significantly increased that number (cf. Figure 4.6). The bounding cuboid with its 20 constraints suffered a selectivity increase of factor 55 compared with the convex hull that produced a selectivity
of roughly 0.06%. Adding all 20 existing inner polytope planes to the cuboid helped reduce the increase to factor 24 while the 1,024 tilted planes resulted in a factor of 15. Using all three approaches together with 1,064 constraints brought the factor down to 11. Compared with that, the greedy point selection using 11, 15, 19, and 23 points out of the convex hull vertices shows that it is a well-suited approach that allows for a very favorable trade-off between the number of constraints and the resulting selectivity. For all further experiments, the box, the inner polytope, and a number of points selected using the greedy heuristic were chosen. As the number of constraints influences the time spent on the preprocessing step, the number of points selected via the greedy heuristic was limited such that no more than 2,000 constraints resulted for the experiments.

All run-time experiments were performed using C++ implementations on a computer with a 2.4GHz Pentium 4 CPU (512 kilobytes of second level cache) with one gigabyte of 266MHz main memory hosting the respective index structure and data points for each measurement.

Using the variance-based heuristic from Section 4.3.2 for deciding the order

Figure 4.10: Comparison of query processing speeds at various reduced dimensionalities
of dropping dimensions, the axis-aligned projection of the Voronoi cell-bounding cuboids results in a definite advantage when indexing the cuboids in an X-Tree. With just four out of the ten dimensions, Figure 4.10 shows the overall nearest neighbor querying speed to double. The Bitvector Index, on the other hand, hardly benefits from said projection. This can be explained by the relatively cheap computation of the final bitvector compared with a high computational cost of both decoding the vector in order to find the relevant data points (where bits are set to one) and computing their distances to the query point. The decoding phase benefits from a sparsely populated bitvector, which necessitates a low selectivity in the filtering step of the nearest neighbor search. The lower number of bit-wise ‘and’ operations that resulted from using only the five dimensions with the highest variances versus all ten dimensions was roughly compensated by the two-fold selectivity increase in the former case.

Figures 4.11 and 4.12 show average response times subject to the projected data dimensionality \( n' \) and the cardinality \( m \) of \( DB \). The linear scan is included as a baseline solution for the nearest neighbor computation. In addition to the two Voronoi-based indexes explained in Section 4.3.3, memory-resident variant
of the X-Tree [BKK96] and the CR-Tree [KCK01] are included as competing approaches using conventional nearest neighbor algorithms. The Priority Search algorithm [HS95] outperformed the Depth-first Traversal algorithm of [RKV95] in all test runs. The latter was thus omitted from the figures. For the 200,000 image histograms of Figure 4.11, all shown indexes outperformed the linear scan over the whole range of examined dimensionalities. Without exceptions, the Voronoi-based Bitvector Index has the lowest querying times followed by the Voronoi-based X-Tree. For the latter, the indexed dimensionality proved optimal at $n^* = 4$ for $n' \in [4, 12]$ and $n^* = 3$ for the rest.

The same order of comparative querying speeds was observed over the whole range of evaluated cardinalities $m$ for fixed $n' = 16$ as depicted by Figure 4.12. Here the point-based hierarchical indexes increasingly outperform the linear scan as the number of data points grows. This is also true for the Voronoi-based X-Tree. Perhaps surprisingly at first, even the Bitvector Index with a linear time complexity displays a seemingly sub-linear behavior. While this is not true in the strict theoretical sense, its slowing increase is explained by the decreasing selectivity that the approximation of the Voronoi cells exhibits with an increasing number of smaller cells. A linear behavior would be expected once the selectiv-

Figure 4.12: Query times at varying cardinality $m$ (dimensionality $n' = 16$)
4.4. Experiments

Figure 4.13: Query throughputs per dataset

Itty closes in on the ideal value of $1/m$ at the latest. At $n′ = 16$ and $m = 200,000$ the speedup factor of the bitmap-based Voronoi approach compared with the next fastest competing approach was over 2.5.

All described indexes and algorithms were evaluated for two more real world datasets in the final experiment depicted in Figure 4.13. In addition to the 200,000 image histograms (taken from TV screen shots), the same number of phoneme-based histograms were extracted from 12,374 spoken and transcribed sentences [Wah00]. A sliding extraction window of one second with a window-to-window overlap of 74% was used to count the occurrences of 27 phonemes. The extraction resulted in approximately 5% non-unique histograms (i.e., windows with identical phoneme counts). Both the image and the phoneme dataset were projected from $n = 27$ to $n′ = 10$ dimensions via PCA. Lastly, 60,000 ratios comparing the per-minute amount of money spent on 10 blue chip stocks were calculated from a subset of the data available in [KF02] and projected to the first six principal components.

In order to visualize the distribution of the resulting histogram vectors in Figure 4.13, all datasets were projected to their first two respective principal components. The hierarchical methods were not as efficient for the latter two
datasets, while the Bitvector Index enabled a vastly increased query throughput for all datasets with a speedup factor in excess of five for the phoneme data.

Acknowledgments
Thanks go to Eammon Keogh and Theodoros Folias for providing access to their Time Series Data Mining Archive [KF02] at the Computer Science & Engineering Department, University of California and to Arne Theres for his work on the phoneme dataset at the Chair of Computer Science 6, RWTH Aachen University.

4.5 Summary
This chapter served as an example of an efficient similarity search system based on a simple distance measure. The efficiency stems from a combination of the techniques reviewed in Chapter 3. The employment of Voronoi diagrams for nearest neighbor search requires special attention regarding the data representation space when reducing the dimensionality of the data from $n$ to $n'$ dimensions. Conservative approximations of Voronoi cells ensure that the correct nearest neighbor for dimensionality $n'$ is found. A hierarchical and a bitmap-based index structure enable the fast computation of a candidate set via point-in-cuboid tests (optionally using a second dimensionality reduction). The nearest neighbor is found among the candidates in the refinement step by computing according distances.

The preprocessing step during which the cell approximations are computed implicitly relies on the Voronoi cells to be of convex shape. This property holds for the Euclidean distance $d_{L_2}$ examined here, but breaks down for other $L_p$-based distances and for more complex distance functions such as the Earth Mover’s Distance examined in Part II of this work. Thus, other techniques for improving the efficiency of query processing are required (cf. Part III).
Part II

Transformation-Based Distance Measures
Compared with the simple distance functions discussed so far, transformation-based distance functions take a different view on defining a distance between two object representations. Instead of measuring the difference between two object representations in a bin-by-bin fashion, they identify the commonality of the representations and assess the amount of work that is required to overcome the remaining differences through transformation of the object representations. Transformation-based distances owe their high degree of flexibility and adaptability to this concept of transformation work. The amount of work that different transformations induce can be flexibly chosen to suit the application requirements. A number of transformation-based distance functions have been proposed in the literature. They are differentiated by the type of data (representation) that they work with, the type of transformations that they allow, and the constraints that may be in place for the transformations. Three transformation-based distance functions that have seen extensive use in similarity search applications will be reviewed in the following chapters.

The Edit Distance [Lev66] for discrete structures is probably the most well-known transformation-based distance. It considers the minimum number of transformation steps (deletions, insertions, and replacements) required to have one discrete structure match the other. While it will not be of importance for the rest of this work, a short introduction to the Edit Distance on discrete-valued sequences will be given in Chapter 5 as it is well-suited to illustrate key concepts of transformation-based distances.

The Dynamic Time Warping distance [Ita75, SC78] is a continuous cousin to the Edit Distance. Instead of deleting or replacing discrete values in a sequence, it effectively replicates existing continuous values and measures the residual differences between the two sequences after an optimal set of replication steps. An introduction to Dynamic Time Warping is given in Chapter 6 with a novel technique for improving its efficiency in a similarity search context in Chapter 13.

The main part of this work is concerned with the efficiency, effectiveness, and applicability of a third transformation-based distance function called the Earth Mover's Distance [RTG98]. It compares feature distributions of two objects in the form of either fixed-binning or adaptive-binning histograms. The details will be introduced in greater depth in Chapter 7.
Chapter 5

The Edit Distance

The Edit Distance [Lev66] is defined by the amount of work that is required to transform a discrete structure into another. The more work that is required for the transformation, the greater the value of the Edit Distance. It was first introduced in [Lev66] for correcting errors in sequences of symbols. Since then, its concept has been extended to several more complex discrete structures. For example, [ZWS96] introduced the Edit Distance for acyclic graphs (i.e., trees) in face of the NP-hard runtime complexity of the Edit Distance on general graphs [BA83].

This chapter illustrates two core concepts of transformation-based distance measures using the Edit Distance on character sequences as an example. This early and well-known transformation-based distance, which is also known as the Levenshtein Distance, has a variety of applications that range from automatic spelling correction [OTK76] to analyzing RNA data in the life sciences [XLS08]. A variation that is used to compare real-valued time series will be discussed in Chapter 6.

5.1 Formal Definition

The Edit Distance between two sequences of symbols (i.e., strings) with generalized transformation costs is recursively defined as the minimum over three Edit Distances calculated for prefixes of the two sequences.
Definition 5.1 *(Edit Distance on Sequences)*

Given an alphabet \( A \), a special symbol \( \diamond \) (empty), and a distance function

\[
gd : (A \cup \{\diamond\}) \times (A \cup \{\diamond\}) \rightarrow \mathbb{R}^+,
\]

the Edit Distance between sequences \( s = \langle s_1, \ldots, s_{m_1} \rangle \) and \( t = \langle t_1, \ldots, t_{m_2} \rangle \) over alphabet \( A \) is defined as

\[
ED_{gd}(s, t) = \min \left\{ \begin{array}{l}
ED_{gd}(s[1 : m_1 - 1], t[1 : m_2 - 1]) + gd(s_{m_1}, t_{m_2}) \\
ED_{gd}(s[1 : m_1], t[1 : m_2 - 1]) + gd(\diamond, t_{m_2}) \\
ED_{gd}(s[1 : m_1 - 1], t[1 : m_2]) + gd(s_{m_1}, \diamond)
\end{array} \right.
\]

with

\[
ED_{gd}(s[1 : 0], t[1 : 0]) = 0
\]

\[
ED_{gd}(s[1 : i], t[1 : 0]) = \sum_{k=1}^{i} gd(s_k, \diamond) \quad \forall 1 \leq i \leq m_1
\]

\[
ED_{gd}(s[1 : 0], t[1 : j]) = \sum_{k=1}^{j} gd(\diamond, t_k) \quad \forall 1 \leq j \leq m_2
\]

where \( u[1 : i] \) stand for the \( i \)-prefix of a sequence \( u \) and \( u[1 : 0] \) for an empty sequence.

Intuitively, the Edit Distance can be understood to transform (w.l.o.g.) the second sequence such that it matches the first one. It is formulated recursively as the minimum over three Edit Distances computed for subsequences that still have to be matched after

1. the substitution of the last symbol of the second sequence,

2. the deletion of the last symbol of the second sequence, or

3. the insertion of a symbol at the end of the second sequence such that it matches the last symbol of the first sequence.

For each of the three transformation options, \( gd \) gives the associated transformation cost where
5.2 Illustration of Core Concepts of Transformation-Based Distances

1. $gd(s_{m1}, t_{m2})$ is the cost of a substitution ($s_{m1}$ instead of $t_{m2}$),

2. $gd(\diamond, t_{m2})$ the cost of deletion (empty instead of $t_{m2}$), and

3. $gd(s_{m1}, \diamond)$ the cost of insertion ($s_{m1}$ instead of empty).

As all three operations result in a shortening of at least one of the sequences, the transformation process always terminates after a finite number of steps. In the end, the Edit Distance identifies an optimal set of transformative operations that results in a match between the two sequences. The value of the Edit Distance is the sum of the costs of the set of optimal transformative operations.

5.2 Illustration of Core Concepts of Transformation-Based Distances

As a simple example, the two names Levenshtein and Löwenstein shall be compared using the Edit Distance with constant edit operation costs of 1 for unequal symbols and 0 for equal symbols. The following table shows how the two names can be aligned using two substitutions (e for ö and v for w) and one insertion:

<table>
<thead>
<tr>
<th>L e v e n s h t e i n</th>
<th>⇕</th>
<th>⇕</th>
<th>⇓</th>
</tr>
</thead>
<tbody>
<tr>
<td>L ö w e n s t e i n</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

This turns out to be an optimal set of transformations and the Edit Distance equals 3 as the number of required transformative operations. Another set of transformations is given by the following alignment.

<table>
<thead>
<tr>
<th>L e v e n s h t e i n</th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Here, the second sequence is transformed to match the first one by inserting all symbols of the first sequence and deleting all symbols in the second sequence. The minimum operator of the Edit Distance causes this non-optimal
Algorithm 5.1: editDistance(s, t, gd)

1. \( m_1 = \text{length}(s); m_2 = \text{length}(t); \)
2. ED = new Array[m1+1, m2+1]; // one extra dimension for empty string
3. ED[1,1] = 0; // edit distance for two empty strings
   // Fill extra row and column (match s, t to empty strings)
4. for \( i = 2 \) to \( m_1 + 1 \) do
   ED[i,1] = ED[i-1,1] + gd(s[i-1], \( \Diamond \));
5. for \( j = 2 \) to \( m_2 + 1 \) do
   ED[1,j] = ED[1,j-1] + gd(\( \Diamond \), t[j-1]);
   // Fill ED matrix for subsequences of s and t
6. for \( i = 2 \) to \( m_1 + 1 \) do
   for \( j = 2 \) to \( m_2 + 1 \) do
   ED[i,j] = \( \min(ED[i-1,j-1] + gd(s[i-1],t[j-1]), \)
   ED[i,j-1] + gd(\( \Diamond \), t[j-1]),
   ED[i-1,j] + gd(s[i-1], \( \Diamond \)));
7. endfor
8. endfor
9. return ED[m1+1, m2+1];

solution with cost 21 to be disregarded. This reflects the first core concept of the transformation-based distances discussed in this work. Transformation-based distance measures investigate a potentially great number of feasible transformations but the final distance value depends on an optimal transformation only. The optimality criterion is based on the second core concept.

Transformation-based distances make use of a measure of transformation cost (typically in the form of a second distance function that is called the ground distance). This allows for great flexibility regarding the modeling of the notion of similarity that is to be reflected by the distance measure. If, for example, the Edit Distance is to be used to model phonetic similarity, the substitution operation for similar-sounding letters such as the labial consonants v and w can be assigned a smaller ground distance. In that case, Levenshtein is judged to be closer to Löwenstein than to Lötenstein (since t is an alveolar consonant).

5.3 Computation

The recursive definition of the Edit Distance can directly be transferred to a recursive algorithm for its computation. However, the number of recursive calls is
5.3. Computation

in $O(3^{m_1+m_2})$. Instead, dynamic programming can be used (cf. Algorithm 5.1) to compute the Edit Distance in $O(m_1 \cdot m_2)$ by avoiding recurring computations of the Edit Distance for identical subsequences.

After computing the non-recursive cases (first row and first column of array $ED$), the algorithm iteratively fills the rest of the array such that $ED[i+1, j+1]$ is the edit distance for subsequences $s[1:i]$ and $t[1:j]$. The value of $ED[i+1, j+1]$ is based on the three adjacent entries given by Definition 5.1. The following table shows the complete array for the earlier example.

<table>
<thead>
<tr>
<th></th>
<th>L</th>
<th>e</th>
<th>v</th>
<th>e</th>
<th>n</th>
<th>s</th>
<th>h</th>
<th>t</th>
<th>e</th>
<th>i</th>
<th>n</th>
</tr>
</thead>
<tbody>
<tr>
<td>L</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>8</td>
<td>9</td>
<td>10</td>
</tr>
<tr>
<td>ö</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>8</td>
<td>9</td>
</tr>
<tr>
<td>w</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>8</td>
<td>9</td>
</tr>
<tr>
<td>e</td>
<td>3</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>8</td>
<td>9</td>
</tr>
<tr>
<td>n</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>3</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>8</td>
</tr>
<tr>
<td>s</td>
<td>5</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
</tr>
<tr>
<td>t</td>
<td>6</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
</tr>
<tr>
<td>e</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>5</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
</tr>
<tr>
<td>i</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>5</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>3</td>
<td>4</td>
<td>5</td>
</tr>
<tr>
<td>n</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>6</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>6</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
</tr>
</tbody>
</table>

The final distance of 3 is stored in $ED[m_1 + 1, m_2 + 1]$.

With the core concepts of transformation-based distances illustrated (the ground distance as a cost measure for transformative operations and the minimum sum of transformations as the overall distance), the next chapter reviews a related measure that is used for real-valued sequences and will be the focus of Chapter 13.
Chapter 6

The Dynamic Time Warping Distance

6.1 Introduction

Time series arise in many different applications that depend on sequential data such as sensor data, stocks data, or videos. For example, large amounts of time-dependent data are created, acquired, and analyzed as a basis for decision and policy-making on ecological issues.

For the scope of this work, a time series is of the form of a feature sequence $t^o = (t_1^o, \ldots, t_m^o)$ where $t_i^o$ describes the features of $o$ at point in time $i$ (cf. Definition 1.3). The analysis of this data requires a notion of similarity between time series to determine like patterns. A simple approach for comparing time series consists of aggregating the differences in values for each point in time (e.g., by using the Euclidean distance $d_{L_2}$). Comparing univariate time series (i.e., $t_i^o$ is scalar) based on Euclidean distance is straightforward. The two time series are interpreted as vectors and differences between values of corresponding points in time are squared and summed up.

$$d_{TS-L_2}(s, t) = \sqrt{\sum_{i=1}^{m} (s_i - t_i)^2}$$
The definition can be generalized to multivariate time series where $t_i$ is a vector.

$$d_{TS-L_2}(s, t) = \sqrt{\sum_{i=1}^{m} d_{L_2}(s_i, t_i)^2}$$

While computationally inexpensive, similarity measures that regard time series in a point-in-time by point-in-time manner are inadequate for many applications. A slight shift in the start time or in the length of a relevant pattern can produce a large distance value and thus a low similarity score. The Dynamic Time Warping (DTW) concept was introduced in the field of speech recognition to address the shortcomings that stem from inflexibility along the time axis [Ita75, SC78] and was transferred to pattern recognition in databases later on [BC94]. The DTW distance computes an optimal alignment of two time series by allowing for stretching and squeezing of the time series. In this manner, it tries to find an assignment of points in time of one time series to points in time in the other time series such that the two time series are as similar as possible to each other after an according transformation of the time axis of one of the time series. DTW then measures the residual difference of the two time series after optimal transformation.

An example for the simple univariate case with a scalar attribute is given in Figure 6.1. Two time series are compared using the Euclidean Distance (a) and DTW (b). The thin lines indicate which values are matched by the respective
distance functions. The vertical extent of the thin lines indicates the amount of
dissimilarity that is contributed by the matching of points in time for the two
models. The Euclidean distance computes a large dissimilarity value between
the two time series even though they show a very similar pattern shifted along
the time axis. DTW matches the time series such that the pattern is better
aligned by transforming or warping the time axis of both time series. Warping
here means that a point in time of one series may be assigned to a different
point in time of the other time series, as long as the assignment is monotonic
(i.e., if \( t_i \) is assigned to \( s_j \), \( t_{i+1} \) may only be assigned to \( s_{j'} \) with \( j' \geq j \)).

The improved handling of the time axis and the possibility to base the cost
of assigning \( t_i \) to \( s_j \) on a ground distance make the DTW a very flexible distance
measure for sequential data. This in turn led to its adoption and adaptation
for diverse application domains going well beyond the speech recognition for
which DTW was originally envisioned. Examples of application domains include
the alignment of gene expressions [AC01], the registration of the time axis of
image sequences [CLC04], the search for similar 2D shapes [KWX+06], and the
detection of near-duplicate videos [AK09, AKS10].

\section{6.2 Formal Definition}

The DTW distance computes the best possible match between two time series
with respect to the overall warping cost given by a ground distance. It is defined
recursively on the length of the sequences.

\textbf{Definition 6.1 (Dynamic Time Warping)}

The Dynamic Time Warping distance between two time series \( s \) and \( t \) of length \( m_1 \)
and \( m_2 \) based on a ground distance \( gd \) is defined as:

\[
DTW_{gd}(s, t) = gd(s_{m_1}, t_{m_2}) + \min \left\{ \begin{array}{l}
DTW_{gd}(s[1 : m_1 - 1], t[1 : m_2 - 1]) \\
DTW_{gd}(s[1 : m_1], t[1 : m_2 - 1]) \\
DTW_{gd}(s[1 : m_1 - 1], t[1 : m_2])
\end{array} \right.
\]
The Dynamic Time Warping Distance

with

\[ DTW_{gd}(s[1 : 0], t[1 : 0]) = 0 \]
\[ DTW_{gd}(s[1 : i], t[1 : 0]) = \infty \quad \forall 1 \leq i \leq m1 \]
\[ DTW_{gd}(s[1 : 0], t[1 : j]) = \infty \quad \forall 1 \leq j \leq m2 \]

where \( t[1 : i] \) stand for the \( i \)-prefix of time series \( t \) and \( t[1 : 0] \) for an empty time series.

Thus, DTW is defined recursively on the minimal cost of possible matches of prefixes shorter by one element – much like the Edit Distance. There are three possibilities:

1. to match the prefixes of both \( s \) and \( t \),
2. to match \( s \) with the prefix of \( t \), or
3. to match the prefix of \( s \) with \( t \).

This definition immediately gives rise to a recursive (but inefficient) algorithm for computing the DTW that computes the assignment cost \( gd(s_{m1}, t_{m2}) \) for the last elements of both time series, recursively computes the three DTW values for shorter time series, takes the minimum of all three DTW values, and finally adds that minimum to the \( gd \) value.

Effectively, the transformation that DTW performs consists of the replication of existing values within the time series. The set of replications that results in the smallest residual difference defines the distance value.

For long time series, infinite warping is typically not desirable. To avoid degenerated matchings where many values of one time series are assigned to very few values of the other one, warping is often restricted via global constraints termed bands. A band describes how much warping is allowed (i.e., how far apart any two assigned points in time can be with respect to the time axis). To disallow a transformation that would leave the feasible band, the ground distance is changed to infinity.
Definition 6.2 (k-band Dynamic Time Warping)
The Dynamic Time Warping distance between two time series $s$ and $t$ of length $m_1$ and $m_2$ (w.l.o.g. $m_1 \leq m_2$) based on a ground distance $gd$ with respect to a bandwidth $k$ is defined as:

$$DTW_{gd,\text{band}_k}(s, t) = gd_{\text{band}_k}(s_{m_1}, t_{m_2})$$

$$+ \min \begin{cases} 
DTW_{gd,\text{band}_k}(s_{[1 : m_1 - 1]}, t_{[1 : m_2 - 1]}) \\
DTW_{gd,\text{band}_k}(s_{[1 : m_1]}, t_{[1 : m_2 - 1]}) \\
DTW_{gd,\text{band}_k}(s_{[1 : m_1 - 1]}, t_{[1 : m_2]})
\end{cases}$$

with

$$gd_{\text{band}_k}(s_i, t_j) = \begin{cases} 
gd(s_i, t_j) & \text{if } |i - \left\lfloor \frac{j}{m_2} \right\rfloor | \leq k \\
\infty & \text{otherwise}
\end{cases}$$

and

$$DTW_{gd,\text{band}_k}(s_{[1 : 0]}, t_{[1 : 0]}) = 0$$

$$DTW_{gd,\text{band}_k}(s_{[1 : i]}, t_{[1 : 0]}) = \infty \quad \forall 1 \leq i \leq m_1$$

$$DTW_{gd,\text{band}_k}(s_{[1 : 0]}, t_{[1 : j]}) = \infty \quad \forall 1 \leq j \leq m_2.$$ 

In this definition, the time warping is restricted to a band of width $k$ in the time dimension by setting the cost of all overstretched assignments to infinity. The band corresponds to a so-called Sakoe-Chiba band [SC78] where the bandwidth is fixed over the whole time series. Another frequently used band [Ita75] allows less warping at the beginning and at the end of the time series.

6.3 Computation

The DTW distance can be computed in $O(m_1 \cdot m_2)$ time via dynamic programming analogous to the computation of the edit distance. Instead of computing all possible alignments, the recursive definition of DTW is used to fill a cumulative distance matrix. An entry at position $(i, j)$ of the matrix corresponds to the best alignment between the prefixes $s_{[1 : i]}$ and $t_{[1 : j]}$ of the time series. Using a $k$-band, the complexity is reduced to $O(k \cdot \max\{m_1, m_2\})$. 
The computation of the matrix entries is illustrated in Figure 6.2, where two time series are depicted to the left (rotated) and at the top of the figure. The optimal assignment of points in time is illustrated by the black line in the matrix. The horizontal segment at the lower left indicates that the first six points in time of the time series at the top are assigned to the first point in time of the time series at the left. With the gray matrix entries representing the band of the DTW, the black path is invalid while the green path is valid. The DTW finds the optimal path from the bottom left to the top right of the matrix using only gray entries. The cumulative distance matrix $D$ is filled analogously to the formula in Definition 6.2.

**Definition 6.3 (Cumulative Distance Matrix)**

The cumulative distance matrix $D \in \mathbb{R}^{m_1 \times m_2}$ for two time series $s$ and $t$ of length $m_1$ and $m_2$ is calculated as

$$D[i, j] = gd(s_i, t_j) + \min\{D[i - 1, j - 1], D[i, j - 1], D[i - 1, j]\}$$

with

$$D[1, 1] = gd(s_1, t_1)$$

$$D[i, 1] = gd(s_i, t_1) + D[i - 1, 1] \quad \forall 1 < i \leq m_1$$

$$D[1, j] = gd(s_1, t_j) + D[1, j - 1] \quad \forall 1 < j \leq m_2.$$
The best matching for two time series is recursively obtained from the matchings that are possible for the time series shorter by one. This means that the entry $D[i, j]$ is computed from its three adjacent entries $D[i - 1, j - 1]$, $D[i - 1, j]$, and $D[i, j - 1]$ to its bottom left (cf. Figure 6.2).

The calculation of entries $D[i, j]$ proceeds column-wise. Starting with the entry at the bottom left (i.e., matrix $D[1, 1]$ – the matrix is vertically mirrored in Figure 6.2 for illustrative reasons), the entries within the band of each column are calculated before continuing with the next column to the right. Global warping path constraint are expressed by setting the matrix entries that are not within the band to infinity. The DTW distance is the value of entry $D[m1, m2]$ in the top right corner. It corresponds to a matching called the warping path. From the cumulative distance matrix, the warping path that corresponds to the minimal matching can easily be constructed as a sequence of matrix cells that correspond to a series of minima that connect the beginnings and ends of the two time series.
Chapter 7

The Earth Mover’s Distance

Both the Edit Distance and DTW are based on the existence of some form of structure in the feature representation. In particular, the Edit Distance for sequences and the DTW distance rely on an inherent total order that allows for a meaningful definition of a prefix for their recursive formulation of the optimization over feasible transformations. The transformation-based distance focused on in this chapter does not require any order for the object features. Instead, it is based on a form of feature representation that includes the adaptive-binning histograms from Section 1.1. Its intuition, application, formal definition, computation, and relevant related work are discussed in this chapter with novel techniques that address its efficiency, effectiveness, and applicability being afforded their own chapters in parts III to V.

7.1 Introduction

The Earth Mover’s Distance (EMD) was first introduced in the field of Computer Vision by Rubner et al. [RTG98, RTG00, RT01] with the intention of assessing the similarity between digital images based on low-level features such as color, texture, and position. It transfers the Monge-Kantorovich mass translocation problem [Mon81, Kan42] to the realm of similarity search by defining a distance measure on the feature representation level (e.g., color histograms) via an optimization-based transformation process. The optimization is formulated
such that it explicitly takes a notion of similarity in the feature space (e.g., color space) into account. In this way the EMD breaks with the assumption of perceptual independence regarding the bins of feature histograms, which distance measures that treat histogram bins in a bin-by-bin fashion are forced to make. An image with a large proportion of pixels falling into the emerald green bin can thus be identified to be more similar to an image with an according portion of kelly green pixels that may have resulted from a slight change in lighting than to an image with largely purple pixels.

The name of the Earth Mover’s Distance stems from the intuition behind its transformation process: Given two feature representations in the form of
7.1. Introduction

Figure 7.3: Movement or flow of earth from hills to valleys

adaptive-binning histograms, one is treated as a collection of hills where the amount of earth that comprises a hill is given by the feature frequency or bin weight and the position of the hill is given by the coordinates of the feature representative in the feature space. For the second histogram, the weights define the depth of corresponding valleys. Figures 7.1 and 7.2 illustrate the concept for an abstract two-dimensional feature space where the two histograms each have four components with distinct representatives (illustrated by the position of the discs in the two-dimensional space) and varying weights (illustrated by the size of the discs in Figure 7.1 and the height/depth of hills/valleys in Figure 7.2). The Earth Mover’s Distance is defined as the least-cost cut-and-fill earth transformation that levels the ground. The cost of moving a unit of earth from a hill to a valley is given by a ground distance measure in the feature space. The more similar the representative that defines the position of a hill is to the representative that defines the position of a valley, the less expensive is the according displacement of earth.

Figure 7.3 shows an example of an EMD computation in a similarity search setting. For the two images in the top left and top right, adaptive-binning histograms with 20 components were determined via a kmeans clustering of pixels in a 5-dimensional spatio-spectral feature space (depicted in the bottom left and bottom right). An optimal transformation is depicted in the middle of the figure.

In its most general definition, the EMD is not restricted to adaptive-binning histograms that share a common total weight of 1.0 but can handle object feature representations with differing total weights (termed signatures in the con-
text of the EMD; cf. Definition 7.1). In that case it performs a partial matching of the earth. If the target signature that defines the valleys in the above example has a lower total weight than the source signature, the EMD finds the transformation that fills all valleys at lowest possible cost. The excess earth remains at its original position. If the source signature has a lower weight than the target signature, a transformation is found that moves all the earth from the positions of the hills to the valleys. Some valleys will not be filled completely.

A number of variants of the EMD have been proposed in the literature. To give an overview on how the concept of the EMD can be adapted to suit differing requirements, a number of them shall briefly be mentioned here; they will not play a role in the rest of this work. In [RT01], a variant called the Partial EMD is introduced together with the Restricted EMD. The former only requires a percentage \( \gamma \) of the signature with the smaller total weight to be matched, which makes it more robust against outliers and missing data. The latter restricts the distance that earth can be moved from any of the hills and measures the percentage of earth that has to remain at its original position under said restriction. A similar idea is followed by the Local EMD [Sri04], where earth may only be moved within a local window around the hills. This restriction can make sense from a perceptual point of view for some applications and can simplify the underlying optimization problem. Unlike Restricted EMD, Local EMD measures the cost of transformation. However, it is not guaranteed that a feasible solution exists in this setting. As a last example for variants of the EMD, the Proportional Transportation Distance [GV02] modifies the EMD such that it obeys the triangular inequality for signatures with unequal total weights. It handles that case by proportionally rescaling the weights of the lighter signature to match the total weight of the heavier signature. In the case of equal-weight signatures (and thus histograms), it is identical to the EMD.

The EMD itself can also be considered a variant of another distance measure. In [LB01] it is noted that the EMD on signatures with equal weight and with a ground distance derived from an \( L_p \)-norm is a special case of the Mallows distance (or Wasserstein distance) from probability theory, which is defined on continuous probability distributions.
7.2 Applications

As the signatures that the EMD takes as its input are of a very flexible nature and comprise both fixed and adaptive-binning histograms and as the ground distance measure in the feature space can be freely chosen, the EMD can readily be adapted to suit a great number of applications. Its relevance to solving real world problems is backed by its successful deployment in fields as diverse as musicology and phishing detection. This section reviews recent applications of the EMD with a focus on medical applications and a brief overview of other application domains.

In the medical field, [DMHE+06] uses the EMD to distinguish personal characteristics from common search patterns in a study on cognitive processes associated with visual search. [LOEA+07] employs the EMD to quantify patterns of cortical behavior associated with surgical expertise based on data derived from multidimensional functional Near Infrared Spectroscopy. Based on CT scans, the EMD is employed to measure the severity of emphysema in [MRM+05] and the progression of advanced cancers in [NC08]. In [EMLB08], parameters of a computational model on chemotaxis-based cell aggregation are iteratively fine-tuned by comparing the results to in vitro experiments using the EMD. [HNZ+08] proposes the use of the EMD to measure discrepancies between simulated and estimated dipole amplitudes for EEG/MEG source reconstruction as it “provides a meaningful measure for arbitrary types of source distributions.” Among many other medical applications, the EMD has also been found to be an effective distance measure for medical image registration [CB07] and for clinical decision-making based on gastroscopic images [WT09], optical coherence tomography [SL09], and multidetector CT [ZBR07]. Evaluating MRI data in breast cancer research, [STN+05] notes that “compared to the euclidean distance a considerably improved classification accuracy could be observed” by using the EMD.

Beyond the medical applications just described and the many applications in the computer vision domain for which it was originally devised (e.g., image retrieval [RTG00, IT03], video retrieval [UHMS06, AKS10], face recognition [LWT05], and image denoising [Kac10]), the EMD has been success-
fully used in many more applications. In [TGV+03, TVW04], notated music is grouped according to melody and authors of anonymous pieces are identified using the EMD. In [LBH98], the EMD is used to compare vector fields that are useful for the modeling of computational and experimental flow fields in physics and engineering. In the field of IT security, the EMD has been used to help identify phishing sites that imitate websites of e-commerce companies in order to gain access to authentication credentials of unsuspecting users [FWD06].

While far from constituting a complete list of applications, the diversity of the aforementioned applications are a good indication of the flexibility and effectiveness of the EMD as a measure of similarity. In their empirical study, the authors of [PBRT99] conclude that “EMD is especially attractive since it allows superior classification and retrieval performance with a much more compact representation.”

7.3 Formal Definition

The feature signatures that the EMD takes as its input are generalizations of adaptive-binning histograms (cf. Definition 1.2 on page 11).

**Definition 7.1 (Feature Signature)**

Given a feature space $FS$ and an object $o$, a feature signature for $o$ is a set

$$s^o = \{(r^o_1, w^o_1), \ldots, (r^o_n, w^o_n)\}$$

of tuples from $FS \times \mathbb{R}^+$ with feature weights $w^o_j$ and feature representatives $r^o_j$.

Notation: Where clear from context, $o$ is used instead of $s^o$ and $|s^o|_1 = \sum_{i=1}^n w^o_i$.

Compared with the adaptive-binning histograms, the only assumption that is made for the feature weights is their non-negativeness. This allows the EMD to perform partial matching. However, much of the remainder of this work will use signatures with a total weight of 1.0.

The EMD itself is defined as a linear optimization problem where linear constraints describe the set of feasible combinations of transformations. The objective function to be optimized is the total cost of transformation. That is, the
EMD determines a feasible combination of transformations such that the total cost of transformation is minimized.

**Definition 7.2 (Earth Mover’s Distance)**

Given two signatures \( s^q \), \( s^o \) and a ground distance \( gd \), the Earth Mover’s Distance between \( s^q \) and \( s^o \) is defined as a minimum over feasible transformations \( F \in \mathbb{R}^{|s^q| \times |s^o|} \):

\[
EMD_{gd}(s^q, s^o) = \min_F \left\{ \frac{1}{\tilde{w}} \sum_i \sum_j F[i, j] \cdot gd(r^q_i, r^o_j) \mid \text{Constraints} \right\}
\]

with

\[
\tilde{w} = \min \{ |s^q|_\Sigma, |s^o|_\Sigma \}
\]

where

\[
\text{Constraints} \equiv \text{CNNeg} \land \text{CSource} \land \text{CTarget} \land \text{CWeight}
\]

defines the feasible transformations with

\[
\begin{align*}
\text{CNNeg} & \equiv \forall i, j : F[i, j] \geq 0 \\
\text{CSource} & \equiv \forall i : \sum_j F[i, j] \leq w^q_i \\
\text{CTarget} & \equiv \forall j : \sum_i F[i, j] \leq w^o_j \\
\text{CWeight} & \equiv \sum_i \sum_j F[i, j] = \tilde{w}
\end{align*}
\]

In the literature, the optimization variable \( F \) is referred to as a *flow matrix* where an entry \( F[i, j] \) stands for the flow of earth or mass from the \( i^{th} \) hill to the \( j^{th} \) valley. The optimal solution consists of a combination of flows that conforms with all constraints and minimizes the overall cost.

The four subsets of constraints define what constitutes a feasible flow in agreement with the intuition of the cut-and-fill process. CNNeg forces all flows to be non-negative as earth is to be translocated from hills to valleys only. A negative value would constitute a flow of “negative earth” or a flow from a valley to a hill as depicted in Figure 7.4(a) where hills are shown in magenta and valleys in blue. CSource and CTarget require that in total no more earth is moved from a hill than is available from that hill and that no more earth is moved to
The Earth Mover's Distance

Figure 7.4: Movement or flow of earth from hills (magenta) to valleys (blue)
a valley than the valley can hold. Assuming that the larger hills/valleys have a height/depth of $2/7$, the flows depicted in figures 7.4(b) and 7.4(c) are in violation of the according constraints. The last constraint CWeight ensures that the signature of lower weight is moved completely. Without the last constraint, the trivial solution of $F[i, j] = 0$ depicted in Figure 7.4(d) minimizes the objective function. Figures 7.4(e) and 7.4(f) show transformations that conform to all constraints. Assuming the Euclidean distance as the ground distance, the former is not optimal and thus dismissed by the EMD. The total cost of transformation is

$$1/7 \cdot \sqrt{50} + 1/7 \cdot \sqrt{10} + 1/7 \cdot \sqrt{45} + 2/7 \cdot \sqrt{4} + 2/7 \cdot \sqrt{2} = 3.4.$$ 

For the optimal case, the total cost of transformation is

$$1/7 \cdot \sqrt{8} + 1/7 \cdot \sqrt{2} + 2/7 \cdot \sqrt{10} + 1/7 \cdot \sqrt{16} + 1/7 \cdot \sqrt{4} + 1/7 \cdot \sqrt{17} = 2.9.$$ 

In the case of adaptive-binning histograms with a total weight of 1.0, the formulation of the EMD can be simplified.

**Definition 7.3 (Earth Mover’s Distance on Adaptive-Binning Histograms)**

Given two adaptive-binning histograms $s^q$, $s^o$ and a ground distance $gd$, the Earth Mover’s Distance between $s^q$ and $s^o$ is defined as a minimum over feasible transformations $F \in \mathbb{R}^{[s^q] \times [s^o]}$:

$$EMD_{gd}(s^q, s^o) = \min_F \left\{ \sum_i \sum_j F[i, j] \cdot gd(r^q_i, r^o_j) \right\} \text{ Constraints}$$

where

$$\text{Constraints} \equiv CNNeg \land CSource \land CTarget$$

defines the feasible transformations with

$$CNNeg \equiv \forall i, j : F[i, j] \geq 0$$

$$CSource \equiv \forall i : \sum_j F[i, j] = w^q_i$$

$$CTarget \equiv \forall j : \sum_i F[i, j] = w^o_j$$
By changing \(C_{\text{Source}}\) and \(C_{\text{Target}}\) to equality constraints, it is ensured that all hills are cut completely and that all valleys are filled completely. In addition, the normalization factor \(1/\tilde{w}\) is not required anymore.

The Earth Mover’s Distance can also be defined for fixed-binning histograms. In that case, the ground distance only has to be defined partially as all query objects and all database objects share the same histogram bins. The partial ground distance is typically represented by a square cost matrix \(C\).

**Definition 7.4 (Earth Mover’s Distance on Fixed-Binning Histograms)**

Given two \(n\)-dimensional fixed-binning histograms \(h^q\), \(h^o\) and a transformation cost matrix \(C \in \mathbb{R}^{n \times n}\), the Earth Mover’s Distance between \(h^q\) and \(h^o\) is defined as a minimum over feasible transformations \(F \in \mathbb{R}^{n \times n}\):

\[
EMD_C(h^q, h^o) = \min_F \left\{ \sum_{i} \sum_{j} F[i, j] \cdot C[i, j] \right\} \quad \text{Constraints}
\]

where

\[
\text{Constraints} \equiv C_{\text{Neg}} \land C_{\text{Source}} \land C_{\text{Target}}
\]

defines the feasible transformations with

\[
C_{\text{Neg}} \equiv \forall i, j : F[i, j] \geq 0
\]

\[
C_{\text{Source}} \equiv \forall i : \sum_{j} F[i, j] = h^q[i]
\]

\[
C_{\text{Target}} \equiv \forall j : \sum_{i} F[i, j] = h^o[j]
\]

Analogously to the feature signature case, vectors of differing total weight can be compared by introducing an otherwise redundant constraint \(C_{\text{Weight}}\), changing \(C_{\text{Source}}\) and \(C_{\text{Target}}\) to inequalities, and normalizing the sum by \(\tilde{w}\).

### 7.4 Computation

Since the EMD is formulated as a linear optimization problem, the same algorithms from the literature that were used to compute Voronoi cell approximations in Chapter 4 can be used to compute the EMD. The formulation allows for
7.4. Computation

a stream-lined simplex algorithm known as the MoDi method or u-v method to be used [HL04]. The outline of the algorithm is as follows:

1. It finds a feasible solution (i.e., an initial $F$ that satisfies all constraints) using for example the North-West corner rule or a more sophisticated techniques such as Vogel’s or Russell’s approximation method described in [HL04].

2. It finds the flow $F[i, j] = 0$ that is not part of the current solution and has the greatest potential for a cost reduction by being included in the solution.

3. If there is no variable with potential for a cost reduction, the solution is optimal and the algorithm terminates.

4. Otherwise, starting at $F[i, j]$, it finds a closed loop of flows that are part of the current solution. Each flow of the loop has to either come from the same source/hill or go to the same destination/valley as the last one.

5. Starting at $F[i, j]$, it adds $\phi$ to every other element of the loop and subtracts it from the remaining elements. The value $\phi$ is chosen such that one element of the loop is reduced to zero.

6. The algorithm returns to step 2.

The analysis of the run-time complexity of the transportation simplex is non-trivial. The simplex algorithm that it is based on is known to have an exponential worst-case complexity on some inputs. However, [ST01] states that “polynomial complexity [is] observed in practice” and that “there has been no satisfactory theoretical explanation of its excellent performance.” Rubner et al. note accordingly in [RTG00] that compared with the exponential worst-case complexity of the simplex algorithm “in practice, because of the special structure in our case and the good initial solution, the performance [of the EMD] is much better.” The authors of [SJ08a] note an “empirical time complexity” between $O(n^3)$ and $O(n^4)$. Experiments in later chapters were performed using an adaptation of the EMD algorithm by Rubner that implements the stream-lined
transportation-simplex algorithm and Russell’s approximation method. The algorithm is in \( \Omega(|s^q| \cdot |s^o|) \) since it has to perform at least one optimality test.

If the Manhattan Distance \( d_{L_1} \) is used as the ground distance and signature representatives are fixed to a small grid, the run-time can be improved using the tree-based technique from \([LO07]\). Alternatively, the transportation problem can also be phrased as a minimum-cost network flow problem with \(|s^q|\) sources, \(|s^o|\) sinks, and \(|s^q| \cdot |s^o|\) edges. This formulation of the EMD can result in improved runtime performance if the ground distance is limited to a set maximum value since this property allows for a reduction of edges in the flow problem \([PW09]\).

### 7.5 Approximations of the EMD

#### 7.5.1 Lower Bounds

As discussed in Section 3.1, lower bounds to a distance measure can be utilized to speed up similarity search query processing. A number of lower bounds to the EMD have been proposed in the literature and shall be reviewed here. The proofs can be found in the according publications.

**Centroid-Based Lower Bound**

A versatile lower bound for the EMD on adaptive-binning histograms with a norm-based ground distance was introduced by Rubner et al. in \([RTG98]\).

**Theorem 7.1 (Centroid Lower Bound for the EMD)**

For a norm-based ground distance \( gd \), the EMD between two adaptive binning histograms \( s^q = \{(r^q_1, w^q_1), \ldots, (r^q_{n_1}, w^q_{n_1})\} \) and \( s^o = \{(r^o_1, w^o_1), \ldots, (r^o_{n_2}, w^o_{n_2})\} \) is lower-bounded by the ground distance between the weighted centroids of the histogram representatives.

\[
LB_{Rubner}(s^q, s^o) = gd\left(\sum_{i=1}^{n_1} w^q_i \cdot r^q_i, \sum_{i=1}^{n_2} w^o_i \cdot r^o_i\right) \leq EMD_{gd}(s^q, s^o)
\]

The lower bound is easy to compute and typically achieves relatively good approximation quality. For fixed-binning histograms, the partial ground distance
information encoded in the transformation cost matrix $C$ does not suffice to apply Rubner’s approximation since the weighted centroids and the ground distance between these centroids need to be defined. Thus, it is required to have access to a fully-defined ground distance $gd$ and feature space partition representatives $r_1, \ldots, r_n$ with $C[i,j] = gd(r_i, r_j)$ to apply $LB_{Rubner}$ on fixed-binning histograms.

**Projection-Based Lower Bounds**

When using $d_{L_1}$ as a ground distance for the EMD, a projection-based lower bound of the EMD can be applied. The idea is to project the representatives of the signatures to a line through the origin and compute the EMD for the projected signatures. A number of projection-based lower bounds are given in [CG97]. Projections onto the feature space axes result in the following bound.

**Theorem 7.2** *(Projection-based Lower Bound for $EMD_{d_{L_1}}$)*

$EMD_{d_{L_1}}$ is lower-bounded by a sum of EMD computation in projections of the $m$-dimensional feature space:

$$LB_{projection-L_1}(s^q, s^o) = \sum_{i=1}^{m} EMD_{d_{L_1}}(s^q[i], s^o[i]) \leq EMD_{d_{L_1}}(s^q, s^o)$$

where $s^q[i]$ and $s^o[i]$ designate projected signatures where all representatives of signatures $s^q$ and $s^o$ are reduced to their $i^{th}$ dimension (combining signature components with common projected representatives).

Instead of computing the EMD for an $m$-dimensional feature space, $m$ 1-dimensional EMD computations are performed. In a 1-dimensional space, $EMD_{d_{L_1}}$ can be solved in linear time, as it is equivalent to the Manhattan distance on cumulative histograms. [CG97] includes a similar lower bound for $gd = d_{L_2}$.

**$L_p$-Based Lower Bounds**

In [AWS06], Assent et al. present three lower bounds to the EMD for fixed-binning histograms. All three bounds are based on weighted $L_p$ distances and
thus have a runtime complexity that is linear in the number of histogram dimensions.

**Theorem 7.3 (Weighted $L_1$ Lower Bound for the EMD)**
For a transformation cost matrix $C$ based on a metric ground distance, the weighted Manhattan distance $LB_{L_1}$ with weights $w_i = \min_{j \neq i} \{C[i,j]/2\}$ is a lower bound to $EMD_C$ on fixed-binning histograms.

$$LB_{L_1}(h^q, h^o) = \sum_i w_i |h^q[i] - h^o[i]| \leq EMD_C(h^q, h^o)$$

**Theorem 7.4 (Weighted $L_2$ Lower Bound for the EMD)**
For a transformation cost matrix $C$ based on a metric ground distance, the weighted Euclidean distance with weights $w_i = \left(\min_{j \neq i} \{C[i,j]/2\}\right)^2$ is a lower bound to $EMD_C$ on fixed-binning histograms.

$$LB_{L_2}(h^q, h^o) = \sqrt{\sum_i w_i (h^q[i] - h^o[i])^2} \leq EMD_C(h^q, h^o)$$

**Theorem 7.5 (Weighted $L_\infty$ Lower Bound for the EMD)**
For a transformation cost matrix $C$ based on a metric ground distance, the weighted maximum distance with weights $w_i = \min_{j \neq i} \{C[i,j]\}$ is a lower bound to $EMD_C$ on fixed-binning histograms.

$$LB_{L_\infty}(h^q, h^o) = \max_i \{w_i |h^q[i] - h^o[i]|\} \leq EMD_C(h^q, h^o)$$

It is shown in [AWS06] that $LB_{L_2}(h^q, h^o) \leq LB_{L_1}(h^q, h^o)$. The authors’ evaluation indicates that the $L_1$ bound outperforms the $L_\infty$ bound regarding its selectivity in a multi-step query-and-refine setup. However, Rubner’s lower bound comes out ahead of the $L_p$-based lower bounds in the majority of experiments.

**Constraint Relaxation-Based Lower Bound**
In addition to the $L_p$-based bounds, [AWS06] also proposes a lower bound based on constraint relaxation, which builds on a minimum of any set being no larger than the minimum of any of its subsets. In the case of the EMD, a superset of
7.5. Approximations of the EMD

feasible flow matrices $F$ is constructed by relaxing one of the constraints.

**Definition 7.5 (Independent Minimization distance $LB_{IM}$)**

Given two $n$-dimensional fixed-binning histograms $h^q, h^o$ and a transformation cost matrix $C \in \mathbb{R}^{n \times n}$, the $LB_{IM}$ distance between $h^q$ and $h^o$ is defined as a minimum over feasible transformations $F \in \mathbb{R}^{n \times n}$:

$$LB_{IM}(h^q, h^o) = \min_F \left\{ \sum_i \sum_j F[i,j] \cdot C[i,j] \mid Cons_{IM} \right\}$$

where

$$Cons_{IM} \equiv CNNeg \land CSource \land CTarget_{IM}$$

defines the feasible transformations with $CNNeg$, $CSource$ as in Definition 7.4 and

$$CTarget_{IM} \equiv \forall i, j: F[i,j] \leq h^o[j].$$

**Theorem 7.6 ($LB_{IM}$ is a lower bound to the EMD)**

The Independent Minimization distance is a lower bound to $EMD_C$ on fixed-binning histograms.

$$LB_{IM}(h^q, h^o) \leq EMD_C(h^q, h^o)$$

In the relaxed optimization problem, the only restriction for the target histogram is that no movement of earth from a single hill may exceed the capacity of a valley. As a result, the optimization can be decomposed into $n$ smaller optimization problems – one for each hill. After an off-line sorting of the rows of $C$, each optimization problem can be solved in linear worst-case time complexity. Experiments in [AWS06] show that $LB_{IM}$ has very good approximation qualities.

The concept can also be generalized to adaptive-binning histograms, where the smaller optimizations have to be solved on-line, leading to a runtime complexity of $O(n^2 \log n)$. For feature signatures of unequal total weight, a global sorting of transformation costs is required.

**Dimensionality Reduction-Based Lower Bound**

Chapter 8 discusses lower bounds based on dimensionality reduction in detail.
A Result on Rubner’s Centroid-Based Lower Bound

When considering Rubner’s Averaging Bound on \( n \)-dimensional fixed-binning histograms together with a Euclidean ground distance between fixed feature space partition representatives \( p_i \in \mathbb{R}^m \), the following equations hold for \( P = [p_1^T; \ldots; p_n^T]^T \in \mathbb{R}^{n \times m} \) as a matrix with \( p_i \) as its rows:

\[
LB_{\text{Rubner}}(h^q, h^o) = d_{L_2} \left( \sum_{i=1}^{n} h^q[i] \cdot p_i, \sum_{i=1}^{n} h^o[i] \cdot p_i \right) \\
= d_{L_2} (h^q P, h^o P) \\
= \sqrt{(h^q P - h^o P)(h^q P - h^o P)^T} \\
= \sqrt{(h^q - h^o)PP^T(h^q - h^o)^T} \\
= d_{QF_{P^T}}(h^q, h^o)
\]

Thus, Rubner’s lower bound turns out to be a quadratic form distance in this case. In fact, it is an optimal lower-bounding quadratic form distance for the EMD as the \( n \)-dimensional ellipsoid of the quadratic form iso-distance surface touches all \( n(n-1) \) vertices of the polytope that describes the EMD on fixed-binning histograms (cf. Chapter 10).

7.5.2 Non-Lower-Bounding Approximations

While approximations of the EMD that are not guaranteed to be lower (or upper) bounds cannot be used in a multi-step filter-and-refine setup without sacrificing the completeness of the result, they can be used as measures of similarity in their own right. Approximations of the EMD have been proposed for instance in [SJ08a], where a metric on wavelet coefficients of a low-dimensional difference histogram is shown to approximate the EMD within loose theoretical and good practical bounds. Based on an approximation technique proposed in [IT03], an approximate grid-based \( L_1 \)-embedding of the EMD is used in [GD04] to match two-dimensional shapes represented by point clouds. Non-lower-bounding approximations will not surface in the remainder of this work as the focus here is on similarity models that are based on the exact EMD.
Part III

Fast Searching with the Earth Mover’s Distance
After reviewing relevant concepts of distance-based similarity search in Part I and of transformation-based distance measures in Part II, the following chapters present original research [WAKS08, AWMS08] on two techniques for improving the efficiency of similarity search query processing for the EMD on fixed-binning histograms.

Chapter 8 describes a technique that takes characteristics of the EMD into account when performing dimensionality reduction on fixed-binning feature histograms. After showing how suitable reductions can be found in a preprocessing step by analyzing the database and/or the EMD and its ground distance, the reduced EMD is utilized as a lower-bounding filter in a filter-and-refine framework.

Chapter 9 defines the MinDist for the EMD, which allows for indexing the EMD in both hierarchical and sequential structures that are based on (minimum) bounding rectangles.

The empirical evaluation of both techniques shows that they are capable of significantly improving the efficiency of the querying process.
Chapter 8

Efficient Search Via Flexible Dimensionality Reduction

The EMD has been successfully adopted in a multitude of applications with low to medium dimensionality. However, multimedia applications commonly exhibit high-dimensional feature representations for which the computational complexity of the EMD hinders its adoption. An efficient query processing approach that mitigates and overcomes this effect is crucial. This chapter proposes dimensionality reduction techniques for the EMD on fixed-binning histograms in a filter-and-refine architecture for efficient and lossless similarity search. An experimental evaluation on real world datasets demonstrates a substantial reduction of the number of expensive high-dimensional EMD computations and thus remarkably faster response times. The techniques are fully flexible in the number of reduced dimensions, which is a novel feature in approximation techniques for the EMD.

8.1 Introduction

The EMD is defined as the minimal amount of work required to change one feature representation into another. To compute the EMD, a linear optimization problem has to be solved. This can be achieved using the simplex algorithm for transportation problems[HL04] as seen in Chapter 7. While the exponen-
tial worst-case complexity is rarely observed in practice, the computation of the EMD is at least quadratic in the feature dimensionality. This is clearly infeasible for high-dimensional feature representations common in multimedia applications. For EMD-based similarity search in this setting, efficient query processing is crucial. Existing multi-step approaches rely on specialized lower-bounding filter functions [RTG98, LBS06, AWS06]. These filters are employed to derive a set of candidates, which are refined using the Earth Mover’s Distance. All of these filter approaches improve the query efficiency without loss of effectiveness but their design is inflexible in the input dimensionality. The approaches in [RTG98] and [AWS06] devise filter distance functions that are limited by the dimensionality of the underlying features space (e.g., 3 for a 3-dimensional color space) or by the dimensionality of the feature representation space (e.g., 64 for 64-dimensional histograms). The filter proposed in [LBS06] only allows for limited flexibility regarding the reduced dimensionality in fixed hierarchical steps of factor 4 and is limited to its grid-based image tiling application domain. Neither of these filtering approaches makes use of information regarding the database at hand.

A different approach for fast EMD retrieval consists of deriving distance functions that serve as approximations of the EMD. Examples of such approximations include [IT03, GD04] where the EMD is embedded into a high-dimensional $L_1$ space with an upper bound for the distortion and [SJ08b] where the EMD is approximated by a sum over wavelet coefficients of the difference histogram. These approaches allow for efficient approximate similarity search but do not guarantee completeness of the retrieval process according to the EMD.

Dimensionality reduction allows for efficiency improvement by means of EMD alone and offers flexibility through choice of the target dimensionality. By reducing the size of the feature representations of the Earth Mover’s Distance, the run-time of the adapted linear program is lowered substantially. For smaller linear programs, average solution times are much lower than for the original problem as the complexity is clearly super-linear. To ensure that dimensionality reduction is a complete filter in the filter-and-refine architecture, the reduced EMD must be a lower bound to the original EMD [KSF+96, SK98].

Techniques for dimensionality reduction of the Earth Mover’s Distance that
8.2 Dimensionality Reduction for the EMD

significantly improve the efficiency of query processing on fixed-binning histograms are proposed in this chapter. The reduction of the EMD is formalized in Section 8.2.1 with proofs for the completeness of the approach and for the optimality of the presented reduced cost matrix in Section 8.2.2. The key to efficient retrieval is then to determine appropriate reductions. A flexible data-independent reduction as a generalization of [LBS06] and a data-dependent method – which incorporates an analysis of EMD transformations – are developed in sections 8.2.3 and 8.2.4. In a filter-and-refine architecture, the EMD at a reduced dimensionality is used in the filter step. The resulting set of candidates is refined using the EMD at the original dimensionality as detailed in Section 8.3. The experimental section 8.4 shows that the proposed approaches improve the query efficiency by producing fewer candidates for the computationally expensive refinement step.

Advantages of the proposed reduction techniques include the reduced computational complexity (efficiency), the arbitrary number of reduced dimensions (flexibility), the possibility of combining them with other EMD lower bounds (chaining) and the absence of false dismissals (completeness).

8.2 Dimensionality Reduction for the EMD

Dimensionality reduction does not rely on separate classes of filter distance functions. Instead, the original distance function is used, albeit on a smaller representation of the features. This is especially helpful for distance functions with super-linear complexity, like the Earth Mover’s Distance. For smaller dimensionalities, distance calculations can be performed in reasonable time. Moreover, dimensionality reduction can be chained with existing filter functions for the EMD. That is, an existing filter can additionally be applied to the reduced data as the result of the dimensionality reduction is again an Earth Mover’s Distance.

For bin-by-bin distances like the $L_2$ norm, a simple form of dimensionality reduction can be devised in a straightforward manner. By discarding dimensions, only non-negative addends are dropped and the resulting distance is guaranteed to be a lower bound. For the Earth Mover’s Distance, discarding dimensions can
result in larger distances since the resulting transformation might be worse than the original one. With a cost of 1 for moving mass one dimension to the right or left, removing dimensions number 2 and 6 for the example in Figure 8.1 results in a greater EMD distance as the cheap flow $F[1, 2]$ can no longer be used. Instead, the much more expensive flow $F[1, 5]$ has to be used. The EMD for $(x, y)$ is $(0.5 \cdot 1 + 0.2 \cdot 0 + 0.3 \cdot 0 + 0.5 \cdot 1)/1.5 = 1.0/1.5$ while the one for $(x', y')$ is $(0.5 \cdot 4 + 0.2 \cdot 0 + 0.3 \cdot 0)/1.0 = 2.0/1.0$. As can be seen, the increase in the EMD is not only a result of the smaller weight normalization (1.0 vs. 1.5). The sum of flows $\times$ weight also increases. Consequently, to avoid false dismissals, simply discarding dimensions is not a valid option for the EMD.

A special case of lower bounds is discussed in [LBS06]. Focusing on bioinformatics image data, twelve separate MPEG-7 color layout descriptor measures are computed for a 12x8 tiling of each image. Each image is associated with 12 separate 96-dimensional feature vectors. For these features, the authors derive a hierarchy of filters, constructed by merging adjacent tiles of the images. The merging of tiles constitutes a special case of linear dimensionality reduction. The reductions proposed in sections 8.2.3 and 8.2.4 are fully flexible in the number of resulting dimensions and have a substantially wider application domain.

As formalized in Definition 7.4, the Earth Mover’s Distance of two $n$-dimensional histograms is defined using an $n \times n$ cost matrix $C$ that reflects the feature space ground distance. Any dimensionality reduction technique for the EMD on
8.2. Dimensionality Reduction for the EMD

histogram vectors thus has to specify two reductions: one for the histogram vectors themselves and one for the cost matrix.

8.2.1 Formalization of the Dimensionality Reduction

Formally, a linear dimensionality reduction from \( n \) to \( n' \) can be described via a matrix.

**Definition 8.1 (General linear dimensionality reduction)**

A general linear dimensionality reduction from dimensionality \( n \) to \( n' \) is characterized by a reduction matrix \( R \in \mathbb{R}^{n \times n'} \). The reduction of an \( n \)-dimensional vector \( x \) to a \( n' \)-dimensional vector \( x' \) is defined as:

\[
x' = x \cdot R
\]

A subtype of linear dimensionality reductions especially useful for the reduction of the EMD are those reductions that combine one or more original dimensions to form a single reduced dimension.

**Definition 8.2 (Combining linear dimensionality reduction)**

The set \( \mathcal{R}^{n,n'} \subset \{0, 1\}^{n \times n'} \) of linear dimensionality reduction matrices that reduce the data dimensionality from \( n \) to \( n' \) by combining original dimensions to form reduced dimensions is defined by:

\[
R \in \mathcal{R}^{n,n'} \iff \forall 1 \leq i \leq n : \sum_{j=1}^{n'} R[i,j] = 1
\]  
(8.1)

Restriction (8.1) together with \( R[i,j] \in \{0, 1\} \) asserts that each original dimension is assigned to exactly one reduced dimension. With \( \{i|R[i,i'] = 1\} \) as the set of dimensions \( i \) that are combined to the reduced dimension \( i' \), it holds that

1. \( \bigcup_{i'=1}^{n'} \{i|R[i,i'] = 1\} = \{1, \ldots, n\} \) (all dimensions are assigned) and

2. \( (i' \neq j') \Rightarrow (\{i|R[i,i'] = 1\} \cap \{j|R[j,j'] = 1\} = \emptyset) \) (no dimension is assigned twice).
Additionally, restriction (8.1) ensures that the reduced vector is of the same total weight as the original vector (\( |x|_\Sigma = |x \cdot R|_\Sigma \)).

A reduced Earth Mover’s Distance is defined via a reduction for the query vector and a reduction for the database vectors. Both reductions are used to compute a reduced cost matrix (cf. Figure 8.2).

**Definition 8.3 (Reduced Earth Mover’s Distance)**

For a cost matrix \( C \), two \( n \)-dimensional vectors \( x \) and \( y \), and two reduction matrices \( R_1 \in \mathbb{R}^{n_1 \times n} \) and \( R_2 \in \mathbb{R}^{n_2 \times n} \), the lower-bounding reduced EMD is defined as:

\[
EMD_{C}^{R_1,R_2}(x,y) = EMD_{C'}(x \cdot R_1, y \cdot R_2)
\]

where \( C' \in \mathbb{R}^{n_1 \times n_2} \) is a lower-bounding reduced cost matrix.

The lower-bounding reduced cost matrix \( C' \) is formally introduced in Definition 8.4. This reduced cost matrix is based on a worst-case assumption to guarantee the lower-bounding property for the filter step. The sparse combining reduction matrices according to Definition 8.2 limit the worst cases that can occur (cf. Section 8.2.2) when compared with dimensionality reduction techniques such as PCA, ICA and Random Projection where \( R[i,j] \in \mathbb{R} \). For example, preliminary tests with PCA (amended by the extra dimension to preserve the total mass) resulted in very poor retrieval efficiency due to the concessions that had to be made for the reduced cost matrix in order to guarantee the lower-bounding property.
8.2. Dimensionality Reduction for the EMD

The two possibly differing reduction matrices $R_1$ and $R_2$ of possibly differing dimensionality applied to the EMD operands (requiring only minor extension of Definition 7.4 to support two differing vector dimensionalities) allow for handling the feature vectors in the database separately from the feature vectors of the queries. In particular, a database reduction to a low dimensionality for indexing in multidimensional structures, and, at the same time, only slight or no reduction of the query for high approximation quality is generally possible with this approach.

8.2.2 Optimal Dimensionality Reduction

In this section, optimality of dimensionality reduction is defined with respect to the efficiency of the similarity search process. During multi-step query processing, dimensionality reduction is used to generate a set of candidates which is refined using the original dimensionality. Smaller candidate sets induce fewer refinement computations and thus result in less computation time for the refinement step. For given target dimensionalities, the optimal dimensionality reduction is therefore the reduction that yields the smallest candidate sets during query processing.

Optimal Cost Matrix Reduction

Any reduction of the dimensionality of the Earth Mover’s Distance requires specification of a corresponding reduced cost matrix. This cost matrix provides the ground distance in the new reduced feature space. Consequently, the reduced cost matrix depends on the reduction matrices of Definition 8.2. The optimal cost matrix with respect to given reduction matrices is the one that provides the greatest lower bound to the EMD in the original dimensionality. As will be proven on the following pages, the optimal reduced cost matrix consists of minima over the original cost entries.

To illustrate why those minima have to be chosen for $C'$, a worst case example is considered: To ensure the lower bound property, underestimating the true distance means assuming the worst case. That worst case occurs if all original mass is transfered at minimum cost by the EMD. For $x = (0, 1, 0, 0)$ and
\( y = (0, 0, 1, 0) \) with a ground distance of \( C[i, j] = |i - j| \), the Earth Mover’s Distance from \( x \) to \( y \) is 1 (moving one unit of mass via flow \( F[2, 3] \) at ground distance \( |2 - 3| = 1 \)). Combining the first two and the last two dimensions, the reduced vectors are \( x' = (1, 0) \) and \( y' = (0, 1) \). The minimum cost entry from the original dimensions 1 or 2 to dimensions 3 or 4 is \( C[2, 3] \), which is indeed the cost that was used in the original EMD. If this value were to be exceeded by \( C'[1, 2] \), the lower bound property would be lost.

**Definition 8.4 (Optimal Reduced Cost Matrix)**

For a cost matrix \( C \), two \( n \)-dimensional vectors \( x \) and \( y \), and a reduced EMD \( C_{R1,R2} \) according to definition 8.3, the optimal reduced cost matrix \( C' \) is defined by:

\[
C'[i', j'] = \min\{C[i, j] \mid R1[i, i'] = 1 \land R2[j, j'] = 1\} \quad (8.2)
\]

In case of \( R1 = R2 \in \mathbb{R}^{n,n'} \), the reduced cost matrix \( C' \) defined by (8.2) is equivalent to the lower bounding cost matrix of [LBS06]. Theorem 8.1 asserts that \( C' \) is lower bounding for \( R1 \neq R2 \), too. Furthermore, Theorem 8.2 states that the reduced cost matrix results in a greatest lower bound for given reduction matrices \( R1 \) and \( R2 \).

**Theorem 8.1 (Lower bound)**

Given two reduction matrices \( R1 \in \mathbb{R}^{n,n1} \) and \( R2 \in \mathbb{R}^{n,n2} \) and a cost matrix \( C \in \mathbb{R}^{n \times n} \), the reduced cost matrix \( C' \) according to (8.2) provides a lower bound for the EMD:

\[
\forall x, y : EMD_{C'}(x \cdot R1, y \cdot R2) \leq EMD_{C}(x, y)
\]

**Proof 8.1**

Let \( \hat{F} \) denote the optimal flow matrix for the original EMD \( C \) and let \( \hat{F}' \) denote the flows combined according to \( R1 \) and \( R2 \):

\[
\hat{F}'[i', j'] = \sum_{\{i | R1[i, i'] = 1\}} \sum_{\{j | R2[j, j'] = 1\}} \hat{F}[i, j]
\]
Then

\[ EMD_C(x, y) = \sum_{i=1}^{n} \sum_{j=1}^{n} \hat{F}[i, j] \cdot C[i, j] \]

\[ \overset{(i)}{=} \sum_{i'=1}^{n_1} \sum_{j'=1}^{n_2} \left( \sum_{[i'R1[i, i'] = 1]} \sum_{[j'R2[j, j'] = 1]} \hat{F}[i, j] \cdot C[i, j] \right) \]

\[ \overset{(ii)}{=} \sum_{i'=1}^{n_1} \sum_{j'=1}^{n_2} \left( \min \{ C[i, j] \mid R1[i, i'] = 1 \land R2[j, j'] = 1 \} \right) \]

\[ \overset{(iii)}{=} \sum_{i'=1}^{n_1} \sum_{j'=1}^{n_2} ( \hat{F}[i', j'] \cdot C'[i', j'] ) \]

\[ \overset{(iv)}{=} EMD_C'(x \cdot R1, y \cdot R2) \]

Step (i) splits the sum over all combinations of \( i \) and \( j \) into sums over the reduced dimensions, making use of each dimension pair \((i, j)\) being assigned to exactly one reduced dimension pair \((i', j')\) by \( R1 \) and \( R2 \).

Step (ii) replaces the individual costs within the brackets by the minimum over all these costs. Hence, the sum can only decrease.

Since this complies with Equation 8.2, it is replaced with \( C'[i', j'] \) in step (iii).

Step (iv) substitutes the summed flows in the brackets with the combined flows introduced above.

Step (v) holds since \( \hat{F}'[i', j'] \) on the left side of the equation is a feasible solution to the transportation problem of the EMD on the right, albeit not necessarily a minimal one.

To help prove the optimality of \( C' \) for given \( R1 \) and \( R2 \), a monotony property of the EMD is introduced. The monotony property ensures that the quality of a lower bound derived via dimensionality reduction increases if the values in the cost matrix increase.
Theorem 8.2 (Monotony of the EMD)
Given two cost matrices $C_1, C_2 \in \mathbb{R}^{n \times n}$ it holds:

$$C_1 \leq C_2 \iff \forall x, y : EMD_{C_1}(x, y) \leq EMD_{C_2}(x, y)$$

where

$$C_1 \leq C_2 \iff (C_1 = C_2) \lor (C_1 < C_2)$$

$$C_1 < C_2 \iff (\forall i, j : C_1[i, j] \leq C_2[i, j]) \land (\exists \hat{i}, \hat{j} : C_1[\hat{i}, \hat{j}] < C_2[\hat{i}, \hat{j}])$$

Proof 8.2
Case 1 ($C_1 = C_2$): The statement trivially holds.
Case 2 ($C_1 < C_2$):

“⇒” Let $\hat{F}$ be optimal for $EMD_{C_2}(x, y) = \sum_i \sum_j \hat{F}[i, j]C_2[i, j]$. Due to $C_2 > C_1$ it holds that $\sum_i \sum_j \hat{F}[i, j]C_2[i, j] \geq \sum_i \sum_j \hat{F}[i, j]C_1[i, j]$ and since $\hat{F}$ is a feasible combination of flows for $EMD_{C_1}(x, y)$, $\sum_i \sum_j \hat{F}[i, j]C_1[i, j] \geq EMD_{C_1}(x, y)$.

“⇐” The proof is given by contradiction: Assume that the statement does not hold (i.e., $C_1 \not\leq C_2$ but still $EMD_{C_1}(x, y) \leq EMD_{C_2}(x, y)$ for all $x$ and $y$), then there exists at least one pair of dimensions $(\hat{i}, \hat{j})$ with $C_1[\hat{i}, \hat{j}] > C_2[\hat{i}, \hat{j}]$. For $(\hat{i}, \hat{j})$, a counter example can be constructed with vectors $\hat{x}$ and $\hat{y}$ where

$$\hat{x}[i] = \begin{cases} 1, & \text{if } i = \hat{i} \\ 0, & \text{otherwise} \end{cases} \quad \text{and} \quad \hat{y}[j] = \begin{cases} 1, & \text{if } j = \hat{j} \\ 0, & \text{otherwise} \end{cases}$$

For these vectors, the assumption does not hold:

$$EMD_{C_1}(\hat{x}, \hat{y}) = 1 \cdot C_1[\hat{i}, \hat{j}] > 1 \cdot C_2[\hat{i}, \hat{j}] = EMD_{C_2}(\hat{x}, \hat{y})$$

After proving the lower bounding property and the monotony of the EMD, the next proof shows that there is no better reduced cost matrix for any given reduction matrices $R_1$ and $R_2$ than the one given by Definition 8.4.

Theorem 8.3 (Optimality)
Given a cost matrix $C \in \mathbb{R}^{n \times n}$ and two reduction matrices $R_1 \in \mathcal{R}^{n, n_1}$, $R_2 \in \mathcal{R}^{n, n_2}$,
there is no greater lower bound than the one provided by \( C' \) according to (8.2):

\[
\neg \exists C'' \in \mathbb{R}^{n_1 \times n_2} \forall x, y : \text{Greater} \land \text{LB} \land (C'' \neq C')
\]

where

\[
\text{LB} \equiv \text{EMD}_{C''}(x \cdot R1, y \cdot R2) \leq \text{EMD}_{C}(x, y)
\]

\[
\text{Greater} \equiv \text{EMD}_{C'}(x \cdot R1, y \cdot R2) \leq \text{EMD}_{C''}(x \cdot R1, y \cdot R2)
\]

**Proof 8.3**

For a proof by contradiction, assume the negation:

\[
\exists C'' \neq C' \forall x, y : \text{Greater} \land \text{LB}
\]

To comply with the Greater constraint, the monotony of the EMD requires \( C' \leq C'' \). Since \( C' \neq C'' \), there must be a pair of dimensions \((\hat{i}, \hat{j})\) with \( C'_{[\hat{i}, \hat{j}]} < C''_{[\hat{i}, \hat{j}]} \). As \( C' \) was computed according to Definition 8.4, it holds that \( C'_{[\hat{i}, \hat{j}]} = \min\{C[i, j] | R1[i, \hat{i}] = 1 \land R2[j, \hat{j}] = 1\} \).

The following two vectors \( \hat{x} \) and \( \hat{y} \) contradict the LB constraint. Let the two original dimensions \( i^* \in \{i | R1[i, \hat{i}] = 1\} \) and \( j^* \in \{j | R2[j, \hat{j}] = 1\} \) be the ones that result in \( C'_{[\hat{i}, \hat{j}]} = C[i^*, j^*] \) and set

\[
\hat{x}[i] = \begin{cases} 1, & \text{if } i = i^* \\ 0, & \text{otherwise} \end{cases} \quad \text{and} \quad \hat{y}[j] = \begin{cases} 1, & \text{if } j = j^* \\ 0, & \text{otherwise} \end{cases}
\]

As the only flow of mass for \( \text{EMD}_{C} \) is between \( \hat{x}[i^*] \) and \( \hat{y}[j^*] \), it follows that

\[
\text{EMD}_{C}(\hat{x}, \hat{y}) = \sum_{i=1}^{n} \sum_{j=1}^{n} \hat{F}[i, j] \cdot C[i, j] = \hat{F}[i^*, j^*] \cdot C[i^*, j^*] = 1 \cdot C[i^*, j^*] = C'_{[\hat{i}, \hat{j}]}
\]

and due to \( i^* \in \{i | R1[i, \hat{i}] = 1\} \) and \( j^* \in \{j | R2[j, \hat{j}] = 1\} \)

\[
\text{EMD}_{C''}(\hat{x} \cdot R1, \hat{y} \cdot R2) = \sum_{i=1}^{n_1} \sum_{j=1}^{n_2} \hat{F}''[i', j'] \cdot C''[i', j']
\]

\[
= \hat{F}''[\hat{i}, \hat{j}] \cdot C''_{[\hat{i}, \hat{j}]} = 1 \cdot C''_{[\hat{i}, \hat{j}]}
\]
which together with $C'[i, j] < C''[i, j]$ contradicts LB:

$$EMD_C(\hat{x}, \hat{y}) = C'[i, j] = EMD_{C''}(\hat{x} \cdot R1, \hat{y} \cdot R2)$$

Theorem 8.3 shows that the reduction of a cost matrix $C$ to $C'$ according to Definition 8.4 is an optimal lower bound for any given reduction matrices $R1$ and $R2$. Therefore, finding good reduction matrices $R1$ and $R2$ is the key to good retrieval efficiency. In the following, only $R1 = R2$ is considered and $EMD_C(x, y)$ is written for $EMD_{C, R}(x, y)$. However, the methods can generally be extended to different simultaneous reductions. In that case, the two matrices should not be chosen completely independently from each other as Definition 8.4 would lead to low values for $C'$. Instead, $R2$ could for example be chosen to be a coarser / finer version of $R1$.

**Optimal Flow Reduction**

As discussed above, the optimal reduction of the cost matrix of the Earth Mover’s Distance depends entirely on the reduction matrices. Consequently, the efficiency of any EMD reduction according to Definition 8.3 depends solely on the choice of the reduction matrix $R$. While it is possible to define what would constitute an optimal choice of $R$ for a given query workload in a multi-step filter-and-refine query processing framework (akin to Section 4.3.2), that optimal $R$ is not attainable in practice. Thus, viable heuristics for finding a suitable $R$ are required. After looking at the optimal $R$ in this section, such heuristics are developed in sections 8.2.3 and 8.2.4.

Given an $n$-dimensional query vector $q$ and a query distance $\epsilon$, an optimal reduction $R \in \mathbb{R}^{n,n'}$ to dimensionality $n'$ can be defined in terms of the number of refinements required to answer an $\epsilon$ range query for a database $DB$. The according characteristic function of the filter (cf. page 46) that uses the EMD in the reduced dimensionality is given by

$$\chi^{R, \epsilon}(q, o) = \begin{cases} 1 & \text{if } (o \in DB) \land EMD^{R}_{C}(q, o) < \epsilon \\ 0 & \text{otherwise.} \end{cases}$$
Due to the lower-bounding property, only elements \( o \) for which \( \chi^{R,\epsilon}(q,o) = 1 \) can potentially still have a refined distance below \( \epsilon \). Since this optimality is only concerned with one single query \( q \), one typically chooses a workload \( Q \) representative of the expected queries and defines optimality with respect to said workload.

**Definition 8.5 (Optimal EMD reduction)**

Given a query workload \( Q = \{(q_1, \epsilon_1), \ldots, (q_t, \epsilon_t)\} \), where \( q_i \) is a query vector and \( \epsilon_i \) the corresponding range threshold, the optimal reduction \( R^* \in \mathbb{R}^{n,n'} \) for \( Q \) is:

\[
R^* = \arg \min_{R \in \mathbb{R}^{n,n'}} \sum_{(q,\epsilon) \in Q} \sum_{o \in DB} \chi^{R,\epsilon}(q,o)
\]

While this equation describes the desired optimal reduction, the search space for the optimization is immense even for small databases and small dimensionalities. Due to the size of the combining reduction matrix, an \((n \cdot n')\)-variable 0-1 integer optimization problem with constraints according to Definition 8.2 has to be solved. Summing over the workload, the objective function consists of \(|Q| \cdot |DB|\) individual \((n' \cdot n')\)-variable EMD optimization problems. Exhaustive enumeration of all possible reductions requires the computation of a total of \(n^n \cdot |Q| \cdot |DB|\) reduced EMD values. Even for a reduction from 16 to 8 dimensions of a database of size 1000 and a workload of size 100, this requires over \(2.8 \cdot 10^{19}\) EMD computations. As this is clearly infeasible even for small databases, heuristics that result in efficient reductions are discussed next.

### 8.2.3 Clustering-Based Reduction

The first approach proposed here is a data-independent dimensionality reduction which is a generalization of [LBS06]. It is motivated by the monotony of the Earth Mover’s Distance (cf. Theorem 8.2).

Monotony induces that a cost matrix with greater entries than another cost matrix is more desirable for dimensionality reduction when used in a filter-and-refine framework. Therefore, this section proposes a method that combines the original dimensions in such a way that the distances between the result-
ing reduced dimensions are as great as possible. At the same time the distance information that is lost shall be as small as possible. These two demands correspond to the well-known clustering goals of maximum inter-class dissimilarity and minimum intra-class dissimilarity.

Figure 8.3 gives an example for \( n = 4 \) and \( n' = 2 \), where the first two original dimensions are combined to the first reduced dimension and the last two original dimensions to the second reduced dimension. The distance that is preserved between the two reduced dimensions is \( C[2,3] = C[3,2] = 2 \), which, according to Definition 8.4, is the minimum of the shaded entries. This inter-class distance should be as large as possible. The distance information \( C[3,4] = C[4,3] = 1 \) within the new second dimension is lost due to the reduction and should thus be as small as possible.

A further postulate for dimensionality reductions is the flexibility in terms of the number of reduced dimensions \( n' \). This flexibility allows full control of the trade-off between the quality of the approximation and the efficiency of the filter step computation. For lower values of \( n' \), the EMD computations in the filter are less expensive to perform but at the same time, the reduced approximation quality results in more refinement computations. The approach discussed here is fully flexible in \( n' \) while the approach of [LBS06] is limited to \( n' = n/4 \) when merging tiles in a hierarchical manner.

The idea that allows for the flexible criterion to be met is to cluster the dimensions based on the ground distance as used by the EMD. Specification of \( n' \)
is possible for partitioning clustering algorithms such as \( k \)-means or \( k \)-medoids [KR90]. Both of these algorithms start with an initial random partition of the data into \( k \) groups, where \( k \) is the user-specified number of clusters. Working in an iterative manner, they assign objects to the nearest cluster center and re-compute these centers for the new partitioning until the clustering is stable with respect to a quality criterion. \( k \)-means uses the arithmetic mean as center of clusters, whereas \( k \)-medoids chooses a central point from the dataset as a cluster representative. The objects in our case refer to the original dimensions of the feature space. As the \( k \)-medoids algorithm does not require an explicit distance function for the feature space but merely a distance matrix between the objects to be clustered, it is chosen over \( k \)-means here. Thus, the dimensionality reduction approach can be applied even if the ground distance function is not explicitly known.

The clustering-based algorithm for finding a reduction matrix starts by randomly choosing \( n' \) representatives (medoids) from the set of original dimensions (i.e., setting one entry in the reduction matrix to 1 for each column with a limit of one non-zero entry per row). It then assigns the remaining dimensions to their nearest medoid according to the cost matrix (i.e., setting a total of \( n - n' \) entries to 1 in the according columns). The quality of the clustering is determined as the total distance based on the reduction matrix \( R \) defined as:

\[
TD = \sum_{i'=1}^{n'} \sum_{\{j R[i,j']=1\}} C[i, i']
\]

The total distance thus reflects the degree of dissimilarity within the clusters (e.g., the distance information that will be lost during reduction) and is the objective function that the algorithm tries to minimize. In the next step, the algorithm aims at improving the clustering. It determines the total distance that results when swapping a non-medoid with a medoid. In a greedy manner, the configuration with the lowest total distance is chosen and the corresponding pair is swapped. The algorithm terminates if no swapping leads to a further improvement of the total distance. The result is a reduction matrix that reflects a clustering of the dimensions into \( n' \) partitions where the partition membership
of the original dimensions is encoded in the columns of the reduction matrix. All original dimensions belonging to a partition are combined to a new reduced dimension.

Since only knowledge about the cost matrix is incorporated in this approach, it is likely that one sacrifices great potential of improving the choice of the reduction matrix. The data-dependent method for dimensionality reduction presented in the next section exploits knowledge that can be derived from the database at hand.

8.2.4 Flow-Based Reduction

The second, data-dependent method for dimensionality reduction introduced here incorporates knowledge on the database at hand to generate a better reduction. It collects flow-information of unreduced EMD computations to guide the process of generating reduction matrices and will thus be referred to as flow-based reduction.

Computing unreduced EMDs to define reduced EMDs for fast retrieval might seem like a paradox at first. However, the unreduced EMDs from which the flow-information is collected are computed in a preprocessing step only and this computational investment is more than justified through faster search times during query processing.

The unreduced EMD is a sum of terms $F[i, j] \cdot C[i, j]$. For a close lower bound, the aim is to achieve largest possible terms $F'[i', j'] \cdot C'[i', j']$ for the reduced EMD. Since the optimal reduced cost matrix $C'$ can be determined by applying Theorem 8.3, the key is to increase the reduced flows with respect to $C'[i', j']$. In this way, the reduced EMDs increase in value and can filter more false candidates in a filter-and-refine setup.

Figure 8.4 illustrates the steps taken to compute a reduction matrix with the flow-based heuristic. In the first step a sample $\mathcal{S}$ is drawn from the database.

Pairwise EMDs (original dimensionality) are calculated on the sample in the second step. The information that is extracted here is the average flow matrix $F^\mathcal{S}$ with

$$F^\mathcal{S}[i, j] = \frac{1}{|\mathcal{S}|^2} \sum_{x \in \mathcal{S}} \sum_{y \in \mathcal{S}} F^{x, y}[i, j].$$
8.2. Dimensionality Reduction for the EMD

Figure 8.4: Process of computing a flow-based reduction matrix

where $\hat{F}^{x,y}[i,j]$ is the flow that occurred from dimension $i$ to dimension $j$ in the optimal solution of the EMD between $x$ and $y$ from some sample $S \subset DB$. Each entry $F^{\varphi}[i,j]$ reflects the average amount of flow from original dimension $i$ to original dimension $j$.

The third and main step of the approach iteratively improves an initial reduction matrix by utilizing the aggregated flow information.

To estimate the quality of a reduction matrix, the flows occurring in reduced EMDs are first approximated by the average original flows $F^{\varphi}$ aggregated according to the reduction matrix that is being investigated.

$$ aggrFlow(F,R,i',j') = \sum_{\{i|R[i,i'] = 1\}} \sum_{\{j|R[j,j'] = 1\}} F[i,j] $$

The aim is to maximize $aggrFlow(F,R,i',j') \cdot C'[i',j']$ in place of the significantly more costly $F'[i',j'] \cdot C'[i',j']$, which would require pairwise reduced EMD computations for each investigated reduction matrix. The experiments in Section 8.4 show that this approximation leads to good results.

Overall, the expected quality of a reduction $R$ is estimated as the sum of the aggregated flows $F^{\varphi}$ weighted by the cost matrix $C'$ optimally reduced according to $R$:

$$ quality(R,F^{\varphi}) = \sum_{i'=1}^{n'} \sum_{j'=1}^{n'} aggrFlow(F^{\varphi},R,i',j') \cdot C'[i',j'] $$
Algorithm 8.1: optimizeFB_MOD(R, C, F, n, n’)

1. origDim = 1; lastOrigDimChanged = 1;
2. currentQuality = calcQuality(R, C, F, 1, R.getAssignment(1), n');
3. repeat
4.   threshold = currentQuality * THRESH; // improvement threshold
5.   for redDim = 1 to n’ do
6.     swapQuality = calcQuality(R, C, F, origDim, redDim, n');
7.     if (swapQuality - currentQuality > threshold) then
8.       R.reassign(origDim, redDim); // change assignment
9.       lastOrigDimChanged = origDim; // track last change
10.      currentQuality = swapQuality;
11.      break; // found improvement, exit for-loop
12.   endfor
13. until (origDim == lastOrigDimChanged);
14. return R;

The global optimization of this term requires computing all possible reductions which is clearly infeasible (cf. Section 8.2.2). Therefore, two variants of hill-climbing algorithms that step-by-step reassign a single original dimension to a new reduced dimension are proposed.

Algorithm 8.1 lists pseudo code for the first proposed variant named FB-Mod (flow-based reduction - modulo). The algorithm takes the current reduction matrix, starts at the first original dimension, and changes its assignment. To this end, it iteratively assesses the assignment of the original dimension to each reduced dimension. If the quality of the resulting reduction matrix is better than the one of the current solution (by some relative margin threshold), the change is made persistent and the algorithm continues with the next original dimension. Once it reaches the last original dimension it starts over at the first one until it visits the same original dimension twice without any reassignments in between.

The expected quality of a reduction matrix that has a single original di-
8.3. Query Processing Algorithm

Algorithm 8.2: calcQuality(R, C, F, origDim, newRedDim, n')

// compute quality of R where origDim is reassigned to newRedDim
1 quality = 0.0;
2 R' = R.copy();
3 R'.reassign(origDim, newRedDim);
4 C' = reduceCostMatrix(C, R');
// sum up the reduced costs times the aggregated flows
5 for i'=1 to n' do
6   for j'=1 to n' do
7     quality = quality + aggrFlow(F, R, i', j') * C'[i',j'];
8   endfor
9 endfor
10 return quality;

mension reassigned to a different reduced dimension is calculated using the calcQuality method that is listed in Algorithm 8.2.

The second algorithm proposed does not necessarily apply the first reassignment that yields a better solution (FB-All of Alg. 8.3). Instead, it evaluates all possibilities before choosing the one single reassignment that results in the best reduction matrix. It then starts the next iteration until no further improvement is achieved.

The remaining building block for the process laid out in Figure 8.4 is the choice of the initial reduction matrix. Two possibilities are proposed here. For a baseline solution, all original dimensions can be assigned to the first reduced dimension. Alternatively, the result from the clustering-based dimensionality reduction (Section 8.2.3) can be used. In this case, the proposed algorithms start from a solution that reflects the ground distances in the feature space. In the experiments of Section 8.4, the two initial reductions are referred to as Base and KMed.

8.3 Query Processing Algorithm

This section describes how a multi-step query-and-refine framework can be forged using the dimensionality reduction technique as a filter step. The focus
Algorithm 8.3: optimizeFB_ALL(R, C, F, d, d')

bestOrigDim = -1; bestRedDim = -1; // track best changes
improved = true;

// get the expected quality of R without any changes
currentQuality = calcQuality(R, C, F, 1, R.getAssignment(1), d');

while (improved == true) do
  improved = false;
  threshold = currentQuality * THRESH; // improvement threshold
  // iterate over all possible reassignments
  for origDim = 1 to n do
    for redDim = 1 to n' do
      swapQuality = calcQuality(R, C, F, origDim, redDim, d');
      // track values if change was an improvement
      if (swapQuality - currentQuality > threshold) then
        currentQuality = swapQuality;
        bestOrigDim = origDim; bestRedDim = redDim;
        improved = true;
      endif
    endfor
  endfor
  if (improved == true) then
    R.reassign(bestOrigDim, bestRedDim); // use best reassignment
  endif
endwhile
return R;

is on nearest neighbor queries, but range queries can be supported similarly.

As described in Chapter 3, complete multi-step query processing requires lower-bounding filter functions [KSF+96, SK98]. Theorem 8.1 guarantees that the proposed dimensionality reduction techniques provide such lower bounds. Consequently, it is possible to integrate the reduced EMD with the algorithms mentioned in Chapter 3. The resulting algorithm for optimal \( k \) nearest neighbor queries [SK98] (in terms of the number of refinements) based on a ranking for the lower-bounding filter function is illustrated in Figure 8.5.

For a specified parameter \( k \) and a query object \( q \), \( k \) initial results are retrieved from the filter ranking. They are refined and inserted into an intermediate result set – sorted with respect to their non-reduced EMD distance from the query \( q \).
8.3. Query Processing Algorithm

Next, the base ranking is queried for the next best object with respect to the filter distance. If the filter distance is smaller than the current $k^{th}$ nearest neighbor in the intermediate result set, the object is refined and compared against the current $k^{th}$ nearest neighbor with respect to the non-reduced EMD distance. If smaller, it is sorted into the intermediate result set, displacing the largest one from the set. This is repeated until the filter distance is larger than the current $k^{th}$ result distance. As soon as the filter distance is larger, none of the remaining objects have a smaller filter distance. And since the filter distance is a lower bound of the non-reduced distance, the non-reduced distance is also larger. The intermediate result set now contains the correct $k$ nearest neighbors.

The dimensionality reduction techniques presented can be flexibly combined. As the reduced distance function again is an EMD computation, it is possible to use existing filters for the EMD on the reduced dimensionality. This chaining of lower-bounding filters allows for efficient query processing as the experiments show. The $LB_{IM}$ technique from [AWS06] is such a lower bound with respect to the Earth Mover’s Distance. In the proposed setup, the ranking Red-EMD required for the k-nearest-neighbor search based on the EMD is thus again based on a ranking – based on $LB_{IM}$ for the objects with reduced dimensionality.

By integrating $LB_{IM}$ in the multi-step setup, a total of three distance functions with increasing computational complexity work together to efficiently find nearest neighbors. Even if no index structure is available to support the ranking queries on the filter level of the k-nearest-neighbor algorithm, only the most
Figure 8.6: Datasets used in experiments. (a) 3,932 feline retina scans, (b) 10,000 radiography images

<table>
<thead>
<tr>
<th>dataset</th>
<th>average EMD</th>
<th>variance</th>
</tr>
</thead>
<tbody>
<tr>
<td>RETINA1-All</td>
<td>1.1998</td>
<td>0.5478</td>
</tr>
<tr>
<td>RETINA3-All</td>
<td>0.3890</td>
<td>0.1951</td>
</tr>
<tr>
<td>RETINA2-All</td>
<td>0.3642</td>
<td>0.1224</td>
</tr>
<tr>
<td>RETINA11-All</td>
<td>0.1241</td>
<td>0.0112</td>
</tr>
<tr>
<td>RETINA7-All</td>
<td>0.0756</td>
<td>0.0011</td>
</tr>
</tbody>
</table>

Table 8.1: Average and variance of EMD distances in the 12 RETINA datasets (sorted by decreasing variance)

An efficient distance function has to be evaluated for all objects of the database. Pseudo-code for the algorithm is given in [WAKS08].

8.4 Experiments

8.4.1 Setup

The reported results in this section are averages over a workload of 100 k-nearest neighbor queries. Each complete dataset (denoted by postfix -All) was divided into a query set (postfix -Q), containing the 100 query objects, and the database (postfix -DB), containing the remaining objects. From the database a sample set was drawn for the data-dependent methods of Section 8.2.4. All experiments were executed on Pentium 4 2.4GHz work stations with 1GB of RAM running Windows XP.
8.4. Experiments

8.4.2 Datasets

The proposed dimensionality reduction approaches are evaluated on two real world datasets referred to as RETINA and IRMA (cf. Figure 8.6). The first image dataset consists of 3,932 feline retina scans labeled with various antibodies and was used in experiments on EMD lower bounds in [LBS06]. For each image, twelve 96-dimensional histograms reflecting a tile-based spatial distribution of twelve MPEG-7 color layout descriptor measures were computed by the authors of [LBS06]. The normalized histograms (sum of 1) result in datasets RETINA1-All through RETINA12-All, from which 100 histograms were each split off for the experiments here to form RETINA1-Q and RETINA1-DB through RETINA12-Q and RETINA12-DB. As RETINA4-All through RETINA12-All exhibited a very low variance in Earth Mover’s distances (cf. Table 8.1), looking for nearest neighbors in those sets is not meaningful. Thus, the experiments are limited to RETINA1-All to RETINA3-All where data-dependent reduction matrices were computed based on 10% samples of RETINA1-DB through RETINA3-DB.

The larger scale experiments use a dataset of 10,000 radiography images from the Image Retrieval in Medical Applications (IRMA) project [LGT+04]. The dataset was part of the 2005 ImageCLEFmed image retrieval competition [DMC+07]. For each image, small patches were collected at both salient points and on a uniform grid. A Linde-Buzo-Gray clustering using the Euclidean distance on 40 principal component coefficients of the patches resulted in 199 cluster centers after applying a threshold algorithm to remove insignificantly small clusters. Details on the feature extraction process can be found in [DKN05]. The dataset IRMA-All comprises the image-wise cluster frequencies stored as 199-dimensional histograms. The 40-dimensional Euclidean distances between the 199 cluster centers are used to compute the cost matrix for the EMD. The data-dependent reduction matrices were calculated on a sample set of 1,000 images derived from the database IRMA-DB, which has a cardinality of 9,900 (i.e., 10,000 images of IRMA-DB minus the 100 query images in IRMA-Q).

8.4.3 Techniques

The following techniques are compared using the architecture from Section 8.3.
• the proposed clustering-based dimensionality reduction (kMedoids)

• the four proposed flow-based dimensionality reduction variants (denoted as FB-All-Base, FB-All-KMed, FB-Mod-Base, and FB-Mod-KMed depending on optimization and initialization)

• the independent minimization filter from [AWS06] (IM)

• the 24-dimensional filter from [LBS06] (Ljosa) which effectively is a lower resolution $6 \times 4$ grid imposed on images from Figure 8.6(a)

• the weighted averaging lower bound from [RTG98] (Rubner)

The lower bound by Ljosa is not applicable for the IRMA database as the 40-dimensional feature space is not organized in a grid-like fashion. The generalized approach from section 8.2.3 (kMedoids) can be applied. As the averaging filter (Rubner) requires a ground distance while the reductions deliver a reduced cost matrix, Rubner can only be applied to the original EMD and is thus run in a direct filter-and-refine setup. All EMD computations are based on $d_{L_2}$ ground distance.

8.4.4 RETINA Experiments

The top left of Figure 8.7 shows that the greatest averaged original “flow $\times$ cost” components in RETINA2 occur in a vertical direction roughly between the two bands of antibody labels that can be seen in Figure 8.6(a). The proposed flow-based reductions keep dimensions separate that induce great “flow $\times$ cost” components in the original EMD while combining those dimensions that contribute low “flow $\times$ cost” components. As can be seen in Figure 8.7, all four variants adapt to the largely vertical flows by assigning different rows to different reduced dimensions. This also keeps the reduced costs between the new dimensions high compared with the kMedoids reduction where each dimension has between 3 and 6 neighboring reduced dimensions with distance 1. The FB-All-Base approach (which has the greatest freedom to find a locally optimal solution) combines a large number of central dimensions to the new dimension
7 as there is little flow originating from this area. The same algorithm initialized with the kMedoids solution shows a more balanced distribution of the new dimensions. For the two modulo variants, little difference can be be seen for the 12-dimensional reduction. They are both rather balanced and adapt well to the vertical flows.

Figure 8.8 shows the 50-nearest neighbor query processing time (on an exponential scale) averaged over datasets RETINA1-DB to RETINA3-DB. The reduced dimensionality is varied for the proposed approaches while the competing approaches are either fixed in their dimensionality (Ljosa) or do not have a reduced dimensionality (sequential EMD without filters, Rubner, and IM). Computing 3,832 expensive high-dimensional EMD distances each for the feature histograms of RETINA1-DB to RETINA3-DB takes on average \( \sim 37 \) seconds.
At $\sim$ 10 seconds, the result of the fixed grid reduction (Ljosa) is a significant improvement. The 3.1s advantage of Ljosa over kMedoids at 24 reduced dimensions may stem from the original feature space partitioning being a regular grid, which is a perfect match for the Ljosa approach. Ultimately, the overall query processing time is of importance. The flexibility of kMedoids in the number of reduced dimensions allows it to outperform Ljosa using 36 reduced dimensions even though it is not optimized for grid-based datasets.

The two direct filter-and-refine approaches (Rubner, IM) give still faster results compared with Ljosa (5.0s and 7.4s respectively) but they are both surpassed by all of the four proposed flow-based approaches at a large majority of reduced dimensionalities evaluated for this experiment (12, 18, 24, 30, and 36). At 2.5 seconds, the FB-All-KMed dimensionality reduction with 18 reduced dimensions shows the lowest overall processing time, closely followed by FB-Mod-Base and FB-Mod-KMed. This means that the flow-based reductions achieve an average response time speedup of factor 15 compared with the sequential EMD and are 2 times faster than the next best competitor (Rubner in this case). The approach with the greatest freedom to choose a reduction matrix (FB-All-Base), which produced a rather large combined area in the middle of the feature space (cf. Figure 8.7), is faster than all data-independent approaches,
Figure 8.9: Relative speedup from applying Red-IM as the first filter chain link

but seems to have overfitted the sample set on which the reduction matrices were computed. The three other variants that produced more balanced reductions are faster in all evaluated configurations. The reduction to 12 dimensions being a less efficient filter is explained by it producing more candidates than the reduction to 18 or more dimensions. This leads to higher overall response times even though the 12-dimensional EMD computations are less costly than the 18-dimensional ones (cf. Figure 8.11 for an experiment regarding this behavior).

Figure 8.9 illustrates the advantage that the reduction-based approaches achieved by including a reduced IM as the first filter chain link according to Figure 8.5. While removing this extra filter step does not change the overall order of approaches seen in Figure 8.8, it is favorable to have this option as it reduces absolute processing times in almost all cases that were studied. The higher the reduced dimensionality, the more it is worth to apply the filter.

When varying the parameter $k$ of nearest neighbors to retrieve (cf. Figure 8.10), the observed order of relative speed remains stable but for one exception. The performance of the IM filter improves significantly when reducing $k$ to very low values. It is consistently faster than Rubner at $k \leq 15$ but matches the proposed flow-based reductions only for $k = 1$. This means that IM was particularly good at approximating distances of very similar histograms. IM re-
turns the nearest neighbor in 0.74s for $k = 1$ where the proposed reduction approaches show response times between 0.81s and 1.06s. Even for $k$ as small as 5, all four flow-based dimensionality reductions outperform all competitors.

### 8.4.5 IRMA Experiments

Figure 8.11 shows the trade-off between the decreasing selectivity and rising processing time of the filter step with increasing dimensionality $n'$. For the larger IRMA dataset with 199 original dimensions, the required number of computationally expensive EMD refinements drops to 2% at a reduced dimensionality of 80 when using FB-All-KMed. At the same time, the superlinear complexity of the EMD causes the share of time spent on the filter step to decrease rapidly for low reduced dimensionalities. The optimum for finding 50 nearest neighbors is achieved at a reduced dimensionality of 60 where roughly 36% of the time is spent on filtering and the remainder of the time is spent on computing high-dimensional EMD refinements for 3.2% of the data. For the same setting, the average filter selectivity of IM is at 21% and at 18.1% for Rubner. The clustering-based reduction from Section 8.2.3 achieves a four-fold selectivity improvement over those two competing results.

Due to the larger number of histograms and the significantly higher dimen-
8.4. Experiments

Figure 8.11: Computation time shares and selectivity of the filter step

For 50 nearest neighbors, Figure 8.12 shows that the two approaches that do not rely on a dimensionality reduction (Rubner and IM) decrease the response times to between 3 and 4 minutes. This is in line with the selectivity values stated above. The proposed data-independent technique (kMedoids) performs similarly well when reducing the dimensionality to between 10% and 15% of the original dimensionality and reaches its optimum of just below 1 minute around dimensionality 60. This value is outperformed by the four proposed flow-based reductions, where FB-All-KMed gives the fastest response time at 46.5 seconds. Again, FB-All-Base performs slightly worse for most dimensionalities than the other flow-based reductions, which are within rather close proximity of each other. Compared with the 17 minutes of the sequential scan and to the 3 minutes of the next best competing approach (Rubner) this equates to a speedup of factor ~22 versus the linear EMD scan and of ~4 versus the next competitor.

For the experiment depicted in Figure 8.13, the IRMA-DB dataset was subsampled in three steps to assess the efficiency of the proposed approaches over a range of database cardinalities. Each of the smaller databases includes the
sample that was taken from IRMA-DB during preprocessing time to compute the flows for the flow-based dimensionality reductions. All approaches scale in roughly the same way and the querying improvements reported above transfer to the datasets at reduced cardinality. From left to right, the relative size of the sample set used for the computation of the aggregated flow matrix decreases from 100% to 10%. The lack of a pronounced superlinear query processing time behavior\textsuperscript{*} indicates that a sample size of 10% suffices to reflect the flows that are worth preserving for this dataset.

In the final set of experiments, the parameter $k$ of nearest neighbors to return was varied (cf. Figure 8.14). Of foremost interest was how the proposed approaches fare compared with IM when $k$ is chosen to be lower than 50. While IM again shows a comparatively low selectivity and a fast average response time, all proposed reduction techniques outperform IM by a factor of at least 1.86.

### 8.4.6 Experiment Summary

The experiments showed that the generalized data-independent dimensionality reduction and the four data-dependent reductions based on original EMD flows

\textsuperscript{*}The $R^2$ error measure for a linear model is close to 0.99 for all four data-dependent methods.
8.4. Experiments

Figure 8.13: Computation time vs. cardinality

are able to outperform competing techniques in a large number of settings. This is in part due to their flexibility regarding the reduced dimensionality, which allows them to make use of the trade-off between decreasing filter selectivity and rising filter processing time. The preprocessing step required for the four flow-based approaches has proven to result in significantly lower query response times, especially for data that exhibits pronounced flow patterns matched by the queries. The reduction matrix derived from the data-independent kMedoids approach showed to be a good starting point for the computation of flow-based reductions. Using this initialization, the −All variant slightly outperformed the −Mod variant in most cases. Speedup factors of up to 22 compared with a sequential scan and up to 4 compared with the next competing approach were achieved.

Acknowledgments
Thanks go to Vebjørn Ljoså, Thomas Deserno, and Thomas Deselaers for providing their datasets.
8.5 Summary

The Earth Mover’s Distance is inherently adaptable to the feature space through its flexible cost matrix. Its computation requires costly minimization of flows between histograms with respect to this cost matrix. In this chapter, novel dimensionality reduction techniques have been proposed for the EMD. The clustering-based reduction relies only on the original cost matrix information to create reduced cost matrices. The approximation quality of the reduced EMD is greatly improved for the flow-based reductions that incorporate EMD flow information from the data. All reduction techniques guarantee completeness in multi-step query processing.
Chapter 9

Direct Indexing and Vector Approximation

This chapter proposes two techniques to enable the indexing of the Earth Mover’s Distance on fixed-binning histograms. Key to both techniques is the development of a MinDist measure that gives the lowest possible EMD distance between a query object and a hyper-rectangle. This measure allows EMD-based retrieval to be supported by a large number of established index structures that organize the feature representation space in rectangles that summarize the database histograms. In addition to hierarchical index structures such as the R-Tree family[Gut84], a lower bound to the MinDist proposed here enables a quantization-based approach to be used for the search process.

9.1 Introduction

With the rapid growth of multimedia data seen in recent years, similarity search in multimedia databases heavily depends on efficient query processing. At the same time, users demand high-quality similarity models. However, high-quality models such as the EMD often induce a high computational cost that contravenes the efficient processing to the point where it turns ineffective. As discussed in Chapter 8, lower bounds such as [RTG98, AWS06, LBS06, WAKS08] can be used to reduce the number of costly refinement computations. Another
method to increase the speed of distance-based similarity search is the exploitation of index structures that help guide the search process. The aim of such index structures is to collect and structure information on the database in a manner that allows for non-relevant parts of the database to be pruned. Well-established index structures that organize the database in such a way include the R-tree family [Gut84], which hierarchically groups database objects in minimum bounding rectangles (MBRs). During query processing, the query histogram is compared with the MBRs in a top-down manner to decide whether the subtrees summarized by an MBR are relevant to the query. In this way, search algorithms based on index structures aim at being selective regarding the input they investigate while pure filter-and-refine frameworks are concerned with the selectivity of their filter output.

For high dimensionalities, the MBRs tend to overlap due to the curse of dimensionality [BGRS99], which forces most MBRs to be examined, requiring random accesses to almost the entire database. Consequently, the sequential scan is eventually faster. This phenomenon has spurred the search for alternative means of indexing. The VA-File from [WSB98] quantizes the data representation space to provide compact representation of features for optimized sequential scans and can be seen as a hybrid between scan-based filter-and-refine and space organizing index structures.

In this chapter, it is shown how indexing based on the VA-File can be employed for the Earth Mover's Distance. As the EMD is intrinsically dimension-interdependent while the VA-File exploits dimension-wise precomputed information, a combination of the two is not trivial. The integration of the EMD query processing with the VA-File is based on a MinDist measure between the query histogram and the fixed grid cells of the VA-File. The efficiency of the query processing is further improved by deriving distance components that go beyond the information stored in a VA-File lookup table without overestimating the true EMD distance.

The MinDist measure formally introduced in Section 9.2 for the first time enables the indexing of the Earth Mover's Distance with a large number of well-established index structures (i.e., those based on bounding rectangles). This type of indexing is referred to as direct indexing as the MinDist is directly based
9.2 Direct Indexing of the EMD

Direct indexing using any indexing technique that relies on bounding rectangles to describe similar feature representation vectors on an abstract level requires the computation of a MinDist measure between a query and the bounding rectangles stored in the index (cf. Figure 9.1). In this chapter, the point-to-rectangle MinDist for the EMD is developed. It enables the direct indexing of the Earth Mover’s Distance using any index based on rectangular bounding regions such as the R-tree family.

**Definition 9.1 (Point-to-Rectangle MinDist for the EMD)**

For a cost matrix $C$, a non-negative $n$-dimensional vector $q$ of weight $|q|_{\Sigma} = 1$, and an $n$-dimensional rectangle with lower and upper boundaries $l$ and $u$ with $|l|_{\Sigma} \leq |q|_{\Sigma} \leq |u|_{\Sigma}$, the point-to-rectangle MinDist for the EMD is defined as

$$MinDist_C(q, l, u) = \min_y \{ EMD_C(q, y) \}$$

Figure 9.1: The EMD MinDist between a vector $q$ and a rectangle defined by $l$ and $u$ is reached at a point $y$ on the rectangle.

on the distance measure that is used in the search (the EMD). In contrast to that, approaches such as [AWS06] index only a lower bound of the EMD or treat the EMD as a black box by using metric indexing techniques [LBS06, CPZ97].

9.2 Direct Indexing of the EMD

For a cost matrix $C$, a non-negative $n$-dimensional vector $q$ of weight $|q|_{\Sigma} = 1$, and an $n$-dimensional rectangle with lower and upper boundaries $l$ and $u$ with $|l|_{\Sigma} \leq |q|_{\Sigma} \leq |u|_{\Sigma}$, the point-to-rectangle MinDist for the EMD is defined as

$$MinDist_C(q, l, u) = \min_y \{ EMD_C(q, y) \}$$
where

\[
\text{Cons}_{\text{MinDist}} \equiv \text{CSum} \land \text{CL} \land \text{CU} \\
\text{CSum} \equiv |y|_{\Sigma} = |q|_{\Sigma} \\
\text{CL} \equiv \forall 1 \leq j \leq n : y[j] \geq l[j] \\
\text{CU} \equiv \forall 1 \leq j \leq n : y[j] \leq u[j]
\]

CSum ensures that only vectors \( y \) of suitable weight are considered while \( \text{CL} \) and \( \text{CU} \) restrict the search for the closest vector \( y \) from \( q \) to the according rectangle. For solving this minimization in practice, the MinDist is restated as a flow minimization and then transformed into a transportation problem. In the end, the MinDist for the EMD has the form of an EMD itself.

**Theorem 9.1 (MinDist as Flow Minimization)**

For \( C, q, l, u \) according to Definition 9.1 and for constraints \( \text{CNNeg} \) and \( \text{CSource} \) according to Definition 7.4, the point-to-rectangle MinDist of the EMD can be stated as

\[
\text{MinDist}_C(q, l, u) = \min \left\{ \sum_{i=1}^{n} \sum_{j=1}^{n} F[i, j] \cdot C[i, j] \mid \text{Cons}_{\text{Flow}} \right\}
\]

with

\[
\text{Cons}_{\text{Flow}} \equiv \text{CNNeg} \land \text{CSource} \land \text{CMin} \land \text{CMax}
\]

\[
\text{CMin} \equiv \forall 1 \leq j \leq n : \sum_{i=1}^{n} F[i, j] \geq l[j] \\
\text{CMax} \equiv \forall 1 \leq j \leq n : \sum_{i=1}^{n} F[i, j] \leq u[j]
\]

**Proof 9.1**

Starting with Definition 9.1, it holds:

\[
\text{MinDist}_C(q, l, u) \\
\overset{0}{=} \min_y \{ EMD_C(q, y) \mid \text{Cons}_{\text{MinDist}} \}
\]
9.2. Direct Indexing of the EMD

\(\text{(ii)}\) \[
\min_y \left\{ \min_F \left\{ \sum_{i=1}^{n} \sum_{j=1}^{n} F[i, j] \cdot C[i, j] \mid \text{Constraints} \right\} \mid \text{Cons}_{\text{MinDist}} \right\}
\]

\(\text{(iii)}\) \[
\min_{y,F} \left\{ \sum_{i=1}^{n} \sum_{j=1}^{n} F[i, j] \cdot C[i, j] \mid \text{CNNeg} \land \text{CSource} \land \text{CTarget} \land \text{CSum} \land \text{CL} \land \text{CU} \right\}
\]

\(\text{(iv)}\) \[
\min_F \left\{ \sum_{i=1}^{n} \sum_{j=1}^{n} F[i, j] \cdot C[i, j] \mid \text{CNNeg} \land \text{CSource} \land \sum_{i=1}^{n} \sum_{j=1}^{n} F[i, j] = \sum_{i=1}^{n} q[i] \right. \\
\left. \land \forall 1 \leq j \leq n : \sum_{i=1}^{n} F[i, j] \geq l[j] \right. \\
\left. \land \forall 1 \leq j \leq n : \sum_{i=1}^{n} F[i, j] \leq u[j] \right\}
\]

\(\text{(v)}\) \[
\min_F \left\{ \sum_{i=1}^{n} \sum_{j=1}^{n} F[i, j] \cdot C[i, j] \mid \text{CNNeg} \land \text{CSource} \land \text{CMin} \land \text{CMax} \right\}
\]

Step (i) is by Definition 9.1 and step (ii) simply expands the definition of the EMD. In step (iii), the two minimizations are combined. In step (iv), the minimization no longer looks for an optimal flow matrix \(F\) given some \(y\) conforming with \(\text{Cons}_{\text{MinDist}}\) but instead looks for the optimal flow matrix that results in a target \(y\) which conforms with \(\text{Cons}_{\text{MinDist}}\). The choice of \(F\) entirely determines \(y\) due to \(\text{CTarget}\). Hence, \(\text{CTarget}\) can be eliminated from the constraints and \(y\) from the minimization search space after replacing \(y[j]\) in all other constraints by the sum of flows going to \(y[j]\) according to \(\text{CTarget}\). In step (v), \(\text{CSum}\) is dropped, as it is a redundant relaxation of \(\text{CSource}\) while \(\text{CL}\) and \(\text{CU}\) become equivalent to \(\text{CMin}\) and \(\text{CMax}\) after the replacement of \(y[j]\).

While also a linear programming problem, this formulation is not in the form of a transportation problem due to the inequalities (9.1) and (9.2). For a transportation problem, which has a favorable structure that allows for more efficient solving, the total weight of both the source and the target must match exactly and all weight from the source must be moved to the target. This can be remedied according to [HL04] by transforming the problem such that each target dimension \(j\) is split into two parts that are each afforded their own target dimension \((n+j)\) in the transformed problem. The first part records how much flow at least has to be sent towards the old target dimension (\(l[j]\)). The second part tracks the amount of flow that can additionally be allotted to the old target dimension (\(u[j] - l[j]\)). The source is extended with the additional
weight required to match the weight of the transformed target.

**Theorem 9.2 (MinDist as a Transportation Problem)**

For $C$, $q$, $l$, $u$ according to Definition 9.1, the point-to-rectangle MinDist of the EMD can be stated as

\[
\text{MinDist}_C(q,l,u) = \min_{F} \left\{ \sum_{i=1}^{n+1} \sum_{j=1}^{2n} F[i,j] \cdot C''[i,j] \mid \text{Cons}_{\text{Transp}} \right\} = \text{EMD}_{C''}(q'',y'')
\]

with

\[
\text{Cons}_{\text{Transp}} \equiv \text{CNNeg''} \land \text{CSource''} \land \text{CTarget''}
\]

\[
C'' = \begin{bmatrix}
C & C \\
\infty & \cdots & \infty & 0 & \cdots & 0
\end{bmatrix}
\]

\[
q'' = (q[1],...,q[n],|u|_\Sigma - |q|_\Sigma)
\]

\[
y'' = (l[1],...,l[n],(u[1] - l[1]),..., (u[n] - l[n]))
\]

where CNNeg'', CSource'', and CTarget'' equal the according constraints of the EMD but for the adapted boundaries for indexes $i$ and $j$ and for the replacement of $q$ and $y$ with $q''$ and $y''$.

**Proof 9.2 (Sketch)**

To obtain a so-called augmented linear program that fits the transportation problem model, inequality constraints are converted to equivalent equalities using slack variables. Slack variables $F[i,j+n]$ are introduced here for assignments that exceed the lower boundaries $l[j]$. The columns of $C$ are doubled in $C''$ as the cost for using a slack variable is to equal the cost of using an original variable. With the last dimension of the source only flowing to slack variables (at no cost), it is ensured that slack variables are only used if the according original variables are already maxed out. The resulting augmented model represents the same solution space as the original one but provides favorable algebraic properties. The transformation to an augmented linear program is discussed in detail in [HL04].

With the theorems from this chapter, fixed-binning histograms can now be managed by MBR-based index structures such as R-Trees. Hierarchical query
processing algorithms [HS95, RKV95] can utilize the MinDist to guide their search progress through the index. Due to Theorem 9.2, it is possible to use the same EMD implementation that is used for determining the distance between two histograms to compute the MinDist of the EMD (but for the weight increase from $|q|_\Sigma$ to $|u|_\Sigma$ which has to be accounted for but is not included in the following section for the sake of readability).

9.3 Vector Approximation for the EMD

In this section, a technique is proposed that reduces the computational overhead during query processing by efficiently precalculating distance components between the query histogram and the MBRs. This requires three properties.

1. The number of upper and lower boundaries of MBRs in the index has to be relatively small,

2. the MinDist computation has to be reduced to dimension-wise distance components, and

3. it has to be possible to efficiently reassemble the dimension-wise distance components to the overall MinDist.

In the next section, the first property is tackled.

9.3.1 Reducing the Number of Boundaries via Quantization

In indexes such as the R-tree, bounding rectangles around feature histograms describe the data. For each of the linear number of leaf nodes in the tree, an MBR is stored in its parent’s node. Whenever a node is searched during query processing, a MinDist computation has to be performed for each of the child nodes. As the MBRs on one level of the tree do not generally coincide in their boundaries, no distance components can be reused for these rectangles. In the following paragraphs, it is demonstrated how distance components can be precalculated once per query based on a priori knowledge of the boundaries in
a quantization approach. This approach could be integrated into a quantized R-Tree but for the sake of simplicity and because of its promised efficiency for high-dimensional feature representation spaces, a non-hierarchical structure (the VA-File [WSB98]) was chosen here.

Quantization maps potentially continuous feature values to fixed intervals in each dimension. Instead of the exact features, bit codes of intervals are stored. Given a real-valued histogram $x$ and a quantization of each dimension $j$ into intervals $b_{j,1}, \ldots, b_{j,m}$ of range $[l_{j,1}, u_{j,1}], \ldots, [l_{j,m}, u_{j,m})$, the number $(k - 1)$ of $b_{j,k}$ is recorded $x[j] \in b_{j,k}$.

In Figure 9.2, an abstract two-dimensional feature representation space is quantized using a regular decomposition with 3 bits, i.e., $m = 2^3 = 8$ intervals. For example, $x = (0.1875, 0.3125)$ in $b_{1,2}$ in the first dimension and $b_{2,3}$ in the second dimension has a quantized representation of $(2 - 1, 3 - 1)$. Its bounding rectangle is given by the corresponding interval ranges $[l_{1,2}, u_{1,2}) = [0.125, 0.250)$, and $[l_{2,3}, u_{2,3}) = [0.250, 0.375)$. In the actual implementation, the quantized representation $(1,2)$ of $x$ is encoded as the concatenated bit string 001010. This explains the shift of $-1$ in the index $k$, as the three bits encode a range of $[0, 2^3 - 1]$. For the sake of simplicity, this encoding and the index shift is ignored in the following sections (i.e., $x$ is quantized to $(2,3)$ instead of 001010.

Using such a quantization technique, the fixed interval boundaries constitute a priori knowledge of possible rectangle boundaries, and their number $n \cdot m$
9.3. Vector Approximation for the EMD

is relatively low and does not depend on the database cardinality. In the VA-File approach, the bit strings are scanned and a MinDist from the query object to the rectangle in which the point falls is computed. The actual distance to the non-quantized object is only computed if the MinDist does not exceed the current distance threshold of, for instance, the range or \( k \)-nearest-neighbor query algorithm. For simple distance functions such as \( d_{L^2} \), the overall MinDist can be reduced to dimension-wise components in the form of \( (q[j] - l_{j,k})^2 \) or \( (u_{j,k} - q[j])^2 \). These components can be computed once per query and stored in a lookup table such that the MinDist for \( d_{L^2} \) is reduced to adding entries of the lookup table based on the bit string of the encoded database object.

However, for the EMD, which globally optimizes flow of mass across dimensions, direct usage of a dimension-wise lookup table is not possible. The next section shows which distance components may nevertheless be stored in a lookup table and how the result from the lookup table can be efficiently augmented with further distance components without sacrificing the completeness of similarity search algorithms.

9.3.2 Lookup-Table Lower Bound

As direct usage of a lookup-table is not possible for the MinDist of the EMD, a lower bound that contains the lookup-table-compatible components for the MinDist of the EMD is proposed here. The idea is to relax the source constraint to obtain a dimension-wise minimization and to only look at target dimensions where it is clear that at least some mass has to be moved from another dimension. Even though the exact mass of a vector in a quantization interval is unknown \textit{a priori}, it is required that mass at least equal to the lower boundary of the interval is moved.

The lookup-table enabled lower bound of the MinDist is defined as follows.

**Definition 9.2 (MinDist lower bound \( LB_{LT} \))**

For \( C^{''} \), CNNeg\(^{''} \), \( q^{''} \), and \( y^{''} \) according to Theorem 9.2, the lookup-table-enabled lower bound \( LB_{LT} \) of the EMD MinDist is defined as a minimization over all possible flows \( F \) under non-negativity constraints CNNeg\(^{''} \), source constraints CSource\(^{LT} \),
and target constraints $C_{\text{Target}}$: 

$$LB_{LT''}(q'', y'') = \min_F \{ \sum_{i=1}^{n+1} \sum_{j=1}^{2n} F[i, j] \cdot C''[i, j] \mid \text{Cons}_{LT} \}$$

with 

$$C_{\text{Source}} \equiv \forall 1 \leq i \leq n+1 : \forall 1 \leq j \leq 2n : F[i, j] \leq q''[i]$$

$$C_{\text{Target}} \equiv \forall 1 \leq j \leq n : (q''[j] < y''[j]) \Rightarrow \sum_{i=1}^{n} F[i, j] = y''[j]$$

The following theorem states that the above definition indeed constitutes a lower bound of the EMD $\text{MinDist}$. 

**Theorem 9.3 (LB$_{LT}$ lower bounds EMD $\text{MinDist}$)**

For $C''$, $q''$ and $y''$ according to Theorem 9.2:

$$LB_{LT''}(q'', y'') \leq \text{EMD}_{C''}(q'', y'') \quad (= \text{MinDist}_C(q, l, u)).$$

**Proof 9.3**

The difference in the definition of $LB_{LT}$ and the $\text{MinDist}$ of the EMD in Theorem 9.2 is the set of constraints. Assuming that the constraints of the lower bound are weaker (i.e., $\{ F \mid \text{Cons}_{LT} \} \supseteq \{ F \mid \text{Cons}_{\text{Transp}} \}$), the theorem would follow:

$$\{ F \mid \text{Cons}_{LT} \} \supseteq \{ F \mid \text{Cons}_{\text{Transp}} \} \overset{(i)}{\iff} \{ F' \in \{ F \mid \text{Cons}_{\text{Transp}} \} \Rightarrow F' \in \{ F \mid \text{Cons}_{LT} \} \} \overset{(ii)}{\iff} \{ F' \in \{ F \mid C_{\text{NNeg}}'' \land C_{\text{Source}}'' \land C_{\text{Target}}'' \} \Rightarrow F' \in \{ F \mid C_{\text{NNeg}}'' \land C_{\text{Source}}_{LT} \land C_{\text{Target}}_{LT} \} \} \overset{(iii)}{\Rightarrow} \min_F \{ \sum_{i=1}^{n+1} \sum_{j=1}^{2n} F[i, j] \cdot C''[i, j] \mid \text{Cons}_{LT} \} \leq \min_F \{ \sum_{i=1}^{n+1} \sum_{j=1}^{2n} F[i, j] \cdot C''[i, j] \mid \text{Cons}_{\text{Transp}} \} \overset{(iv)}{\iff} LB_{LT''}(q'', y'') \leq \text{EMD}_{C''}(q'', y'')$$
9.3. Vector Approximation for the EMD

Step (i) holds by definition of the subset relation and step (ii) by definition of \(\text{Constransp}\) and \(\text{Cons}_{LT}\). Step (iii) follows as the minimum over a subset cannot be smaller than the minimum over the whole set. Step (iv) is true by definition of \(\text{LB}_{LT'}\) and \(\text{EMD}_{C''}\).

For the whole theorem to hold, one of the first three statements has to be proven. If \(\{F \mid C_{\text{Source}}''\} \subseteq \{F \mid C_{\text{Source}}_{LT}\}\) and \(\{F \mid C_{\text{Target}}''\} \subseteq \{F \mid C_{\text{Target}}_{LT}\}\) then the third statement holds and thus the whole theorem holds, too.

Looking at the definition of \(C_{\text{Source}}'' \equiv \forall 1 \leq i \leq n+1 : \sum_{j=1}^{2n} F[i,j] = q[i]\) and \(C_{\text{Source}}_{LT}\) from above, the first inclusion holds because requiring that a sum of non-negative elements \(F[i,j]\) equals \(q[i]\) is a stronger constraint than requiring that each element of the sum is at most equal to \(q[i]\).

As for \(C_{\text{Target}}'' \equiv \forall 1 \leq j \leq 2n : \sum_{i=1}^{n+1} F[i,j] = y''[j]\), the constraint \(C_{\text{Target}}_{LT}\) simply limits fewer of the columns in \(F\) in the same way \(C_{\text{Target}}''\) does.

The main advantage of \(\text{LB}_{LT'}\) over the MinDist is that the optimization can be performed per column of the flow matrix \(F \in \mathbb{R}^{(n+1) \times 2n}\). For columns \(j\) with \(j > n\) or \(q''[j] \geq y''[j] = l[j]\), the only restriction is that the entries \(F[i,j]\) are in the interval \([0, q''[i]]\). A choice of 0 trivially minimizes the sum. For the remaining columns \((j \leq n\) and \(q''[j] < y''[j] = l[j]\)), an amount of mass equal to \(y''[j] = l[j]\) has to be distributed to entries \(F[i,j] \in [0, q''[i]]\). Unlike for \(\text{EMD}_{C''}\), a choice for a value \(F[i,j]\) in such a column does not influence optimal choices outside that column and the order of optimal assignment within the column only depends on the cost matrix \(C''\).

The computation of the complete lookup-table \(LT\) for a quantization with lower quantization boundaries \(\text{LBoundaries}\) is given in Algorithm 9.1. If \(q[j]\) is less than the \(k^{th}\) lower boundary value \(\text{LBoundaries}[j][k]\) in dimension \(j \leq n\), mass equal to the lower boundary is successively moved to dimension \(j\) from dimensions given by the ordering of source dimensions in the input parameter \(\text{ColSorting}[j]\). The value of \(\text{ColSorting}[j][i]\) is the row index of the \(i^{th}\) smallest cost entry in column \(j\) of \(C\). If not enough mass is available from one source, the next more expensive source is chosen. The algorithm has a run-time complexity of \(O(n^2m)\) for \(m\) lower boundaries in \(n\) dimensions.

After computing \(LT\) once for a query \(q\), values \(\text{LB}_{LT} \leq \text{MinDist}(q,o)\) for
Algorithm 9.1: computeLT(C, q, ColSorting, LBoundaries)

1. for $j = 1$ to $n$
   2.   for $k = 1$ to $m$
      3.      // Calculate minimum sum of flows $\times$ costs for $k^{th}$ lower boundary in the $j^{th}$ dimension
      4.      dist = 0;
      5.      if ($q[j] < LBoundaries[j][k]$) then
      6.         mass = LBoundaries[j][k];
      7.         for $i = 1$ to $n$
         8.            // iterate over the $j^{th}$ column of $C$ in ascending cost order
         9.            nextsource = ColSorting[j][i];
        10.            smallermass = min($q[nextsource]$, mass);
        11.            dist = dist + smallermass $\times$ C[nextsource, j];
        12.            mass = mass - smallermass;
        13.      endfor
      14.      endif
    15.   endfor
  16. return LT;

$o \in DB$ can be computed with simple table lookups in $O(n)$ time as shown by Algorithm 9.2 where approx\_o is the quantized representation of the $n$-dimensional database object $o$.

The range query performed by Algorithm 9.3 shows that $LT$ is only computed once per query $q$ and that ColSorting is independent of $q$. In addition to the parameters of the rangeQ algorithm (cf. page 20), the sorting of cost entries in $C$ and the VA-File with quantization boundaries and vector approximations are required as inputs. The sorting of cost entries can be performed offline in $O(n^2 \log n)$ for $n$ dimensions. The lower bound to the MinDist of the EMD can similarly be integrated into the two $k$-nearest-neighbor algorithms VA-SSA and VA-NOA given in the technical report [WB97] that details the techniques presented in [WSB98].
9.3. Vector Approximation for the EMD

Algorithm 9.2: computeLBLT(approx_o, LT)

1. sum = 0;
2. for j = 1 to n do
3.     sum = sum + LT[j][approx_o[j]];
4. endfor
5. return sum;

Algorithm 9.3: LB_LT-rangeQ(q, ε, EMD_{C}, DB, ColSorting, VA)

1. ResultSet = ∅;
2. LT = computeLT(C, q, ColSorting, VA.LBoundaries);
3. for i = 1 to |DB| do
4.     if computeLBLT(VA.approximations[i], LT) ≤ ε then
5.         if EMD_{C}(q, DB[i]) ≤ ε then
6.             ResultSet = ResultSet ∪ {DB[i]};
7.         endif
8.     endif
9. endfor
10. return ResultSet;

9.3.3 Optimization of the Lookup-Table Lower Bound for Metric EMD

The quantization approach up to here did not make use of any properties of the ground distance (i.e., there is no restriction on C). However, for the popular class of metric ground distances, an optimization can be made as some of the flows for $EMD_{C''}(q'', y'')$ are known. In particular, the definiteness of a metric ground distance results in a matrix $C''$ of the form

$$C'' = \begin{bmatrix} 0 & \cdots & 0 \\ \cdots & \ddots & \cdots \\ 0 & \cdots & 0 \end{bmatrix}$$

with non-zero entries in all other positions.

An improved lower bound for metric EMDs can be achieved by considering
the mass that the lower bound did not yet take into account. According to the constraint $C_{\text{Target}_{LT}}$, only those target dimensions $j < n$ for which $q[j] < l[j] = y''[j]$ are taken into account by $L_{LT}$. The amount of mass that is distributed to such target dimensions $j$ by $L_{LT}$ is the same as the mass distributed by $EMD_{C''}$; however, $L_{LT}$ may move the mass at lower cost due to having fewer restrictions. For all other columns of $F$, a flow of 0 is allowed by $L_{LT}$ wherever $C''$ is greater than 0. For $EMD_{C''}$, this is not the case. The cost of moving the mass that occurs in those other columns for $EMD_{C''}$ can be conservatively estimated by distributing the mass in ascending cost order similar to the mass distribution starting in line 5 of Algorithm 9.1. To determine the amount of mass that is transferred at cost by the EMD and not yet accounted for by $L_{LT}$, it is helpful to divide the dimensions of $q$ into three cases illustrated in Figure 9.3. The first case is already accounted for by $L_{LT}$ with a mass of $y''[i] = l[i]$ in column $i$ of $F$ due to $C_{\text{Target}_{LT}}$. In the second case, the EMD moves $l[i]$ via
9.3. Vector Approximation for the EMD

$F[i, i]$ and $q[i] - l[i]$ via $F[i, i + n]$ for a total of $q[i]$ at cost zero. In the third case, the EMD moves $l[i]$ via $F[i, i]$ and $u[i] - l[i]$ via $F[i, i + n]$ for a total of $u[i]$ at cost zero. Since the slack dimension $q''[n + 1] = |u|_\Sigma - |q|_\Sigma$ is moved at zero cost, the sum of mass accounted for by $LB_{LT}$ equals

$$\sum_{q[i]<l[i]} l[i]$$

and the sum of unaccounted mass that is moved by the EMD at zero cost is

$$\left(\sum_{l[i] \leq q[i] < u[i]} q[i]\right) + \left(\sum_{u[i] \leq q[i]} u[i]\right) + (|u|_\Sigma - |q|_\Sigma).$$

With the total mass moved by the EMD being $|u|_\Sigma = \sum_{i=1}^n q[i] + (|u|_\Sigma - |q|_\Sigma)$, the amount of mass that is transferred at cost by the EMD and not yet considered by $LB_{LT}$ is equal to

$$\Delta = \sum_{q[i]<l[i]} (q[i] - l[i]) + \sum_{u[i] \leq q[i]} (q[i] - u[i]).$$

The last task left is to conservatively estimate at what cost the mass $\Delta$ can be moved without jeopardizing the lower bounding property. By adding that cost to $LB_{LT}$, a closer lower bound of the MinDist for the EMD is established. The general idea is to assume that as much mass as possible moves at as low cost as possible. This can be achieved with a fast greedy algorithm that assigns mass from dimension $i$ to $j$ in the order of ascending entries of $C[i, j]$ where entries that will not be used for this mass by the EMD can be excluded. In the third case of Figure 9.3, it is clear that the target constraints are already satisfied for dimensions $i$ and $i + n$. Thus, the $i^{th}$ column of $C$ can be ignored in that case. It is also clear from the discussion above that the mass does not flow at zero cost. The amount of mass that the greedy algorithm may move from dimension $i$ to other dimensions can be limited to $q[i]$ while the amount of mass being moved to dimension $j$ is left unrestricted to allow for a greedy solution algorithm.

As the distribution algorithm depends on its input $\Delta$ in the form of a sum over differences of $q$ minus the quantization boundaries, this technique is not
Algorithm 9.4: \( \text{LB}_{LT}^{+}-\text{rangeQ}(q, \epsilon, \text{EMD}_C, \text{DB}, \text{ColSorting}, \text{EntrySorting}, \text{VA}) \)

1 ResultSet = \(\emptyset\);
2 LT = computeLT(C, q, ColSorting, VA.LBoundaries);
3 for \(i = 1\) to \(|DB|\) do
4    LB_LT = computeLBLT(VA.approximations[i], LT);
5    if \(LB_LT \leq \epsilon\) then
6       if \(LB_LT + \text{greedyMassAssignment}(q, C, VA.LBoundaries, VA.UBoundaries, \text{EntrySorting}) \leq \epsilon\) then
7          if \(\text{EMD}_C(q, \text{DB}[i]) \leq \epsilon\) then
8             ResultSet = ResultSet \(\cup\) \{DB[i]\};
9          endif
10     endif
11    endif
12 endfor
13 return ResultSet;

compatible with the lookup-table but has to be computed as an optimization for every pair of \(q''\) and \(y''\) (cf. Algorithm 9.4). However, for high-dimensional EMD computations, the additional computational overhead pales compared with the EMD and pruning a few more percent points of EMD computations is well worth the effort.

9.3.4 Multistep Algorithm

The lookup table lower bound \(LB_{LT}\) proposed allows for efficient query processing in a nearest neighbor multi-step algorithm [SK98]. As illustrated in Figure 9.4, the lower bound \(LB_{LT+}\) (i.e., \(LB_{LT}\) on the lookup table plus the online optimization) is used as a filter on the database to quickly generate a small set of candidates. Additionally, \(LB_{IM}\) further reduces this candidate set. Finally, EMD refinement ensures that the correct result set is obtained. Since the proposed lower bounds underestimate the true EMD MinDist, completeness in multi-step processing is ensured.

The \(k\)-nearest-neighbor query processing algorithm used in the evaluation section begins by computing a ranking according to the lower bound \(LB_{LT}\) of the
EMD MinDist. This ranking ensures that the number of computations in query processing is minimized, as discussed in [SK98]. More precisely, a ranking according to the filter distance is used to compare the next best filter candidate against the worst refined distance result so far. As soon as the filter distance exceeds the worst refined neighbor distance in the intermediate result set, the set of nearest neighbors is complete. Any of the following candidates are guaranteed to have higher filter distances and thus higher true EMD distances than the set of nearest neighbors already retrieved. Consequently, they can be safely pruned.

The ranking is computed using an efficient sequential scan on quantized histograms of the compact VA-File [WSB98]. Starting with the nearest neighbors according to this filter distance, exact representations of the histograms are used to compute the $LB_{IM}$ lower bound from [AWS06] and the EMD is only computed if a candidate cannot be pruned by either filter.

9.4 Experiments

9.4.1 Setup and Datasets

The main experiments in this sections use the database of 200,000 color TV images from Section 4.4. For a random sample of the images (size 10,000), a random sample of pixels (size 500) was selected. For those pixels, relative x and
y position, the three color features of the CIE Lab model, and local texture measures (contrast and coarsity) were computed and normalized according to their standard deviation. The resulting five million seven-dimensional feature vectors were clustered using k-means to determine a given number of cluster centers, which served as partition representatives for the subsequent fixed-binning histogram computation of all images. The Euclidean distance between those centers was used to compute the cost matrix for the EMD. A second image dataset consisted of 50,000 heterogeneous digital photos and was processed in the same manner. For each set, 50 separate images were selected as query images. All experiments were executed on Pentium 4 2.4GHz work stations with 1GB of RAM where a warmed up system state was simulated by posing 25 queries and then recording the query processing times for next 25 queries.

### 9.4.2 Techniques

Besides the VA-File using the proposed MinDist lower bound $L B_{LT}$ for a regular 4 bit quantization grid, the EMD-based MinDist from Section 9.2 was utilized for the direct indexing of the EMD in an R-Tree with a node block size of 16kb. The index selectivity was computed as the percentage of objects that had to be loaded from the original database by the indexing structure. For the VA-File with its ranking based nearest-neighbor search, this measure equates to the relative number of times that a refinement step on the non-quantized data is required. For the R-Tree, the selectivity is the number of data points read in a leaf nodes divided by the database size. In addition, a linear scan over the database was performed as a baseline that loaded all database objects. In all three cases, the expensive EMD in the refinement step is preceded by an $L B_{IM}$ filter computation.

### 9.4.3 Results

In the first set of experiments, the database size was fixed to 100,000 data points and 10 nearest neighbors were computed for dimensionalities between 5 and 100. As was expected and is shown in Figure 9.5, the selectivity of the R-Tree
Figure 9.5: Dimensionality vs selectivity

drops rapidly and crosses the 50% line at around 10 dimensions. At the same time, the VA-File with \( LB_{LT+} \) manages to maintain a very good index selectivity over the complete range of dimensionalities.

The low number of data points that have to be examined by the \( LB_{IM} \) in the refinement step leads to a low number of expensive EMD computations that have to be computed in the end. Figure 9.6 shows that out of 100,000 data points, on average no more than 70 extra EMD computations have to be performed by the VA-File approach to find the 10 nearest neighbors. Even though the R-Tree does not prune many data points from the search space for high dimensionalities, it still manages to outperform the linear scan in the number of EMD computations as the order of accessing the data nodes determined by the MinDist guides the search to more quickly find database objects with small EMD values. This in turn results in more candidate dismissals by the \( LB_{IM} \) filter in the following steps. However, the R-Tree uses additional expensive MinDist computations during the search.

Figure 9.7 shows the average query time for a 10-nearest-neighbor search with a logarithmic scale. Up to a dimensionality of ten, the R-Tree exhibited the by far fastest response times, making it a top choice for the low dimen-
sional case. For higher dimensionalities, the random accesses, MBR overlaps, and increasingly expensive MinDist computations during the directory traversal make the R-Tree performance decrease rapidly. Contrary to that, from dimensionality 10 onwards, the VA-File runs about one order of magnitude faster than
the linear scan utilizing $LB_{IM}$, making it the clear favorite for the medium to high-dimensional case.

For the experiment shown in Figure 9.8, the database cardinality was varied with a fixed dimensionality of 50. As discussed before, the R-Tree can not compete in this 50-dimensional setting. The VA-File on the other hand scales very well. Its slightly super-linear behavior can be explained by the logarithmic time operations on the larger heap it maintains while computing rankings according to $LB_{LT+}$.

In order to see whether the behavior of the three approaches was highly specific to the TV dataset where some dense clusters exist (scenes with long lasting backgrounds), a subset of the experiments were performed on a more heterogeneous database consisting of a web collection of user-uploaded digital photos with a large variety of categories. As Figure 9.9 shows, there was hardly any difference in behavior for otherwise identical settings. Some additional experiments on the parameter $k$ of objects to be retrieved and on the database cardinality can be found in [AWMS08].
9.5 Summary

In this chapter, a point-to-rectangle MinDist measure was provided for the EMD. It is a key prerequisite for the integration of the EMD in any index structure based on multi-dimensional bounding rectangles. Moreover, an optimized quantization-based indexing technique based on the MinDist and the VA-File was proposed. Devising a lookup-table and a conservative approximation of remaining costs, the search for similar objects in a database can be significantly sped up as shown by the experiments. The completeness of the multi-step algorithm is proven by showing that the filter is indeed a lower bound of the EMD MinDist.
Part IV

Effective Searching with the Earth
Mover’s Distance
The ground distance that defines the transformation cost within the feature space for the EMD is the focus of original research [BWS09, WBSS09a, WBSS09b] that is presented in Part IV of this work. The adaptability of the ground distance offers an opportunity to improve the effectiveness of similarity search tasks and allows for custom fitting of the EMD to user or application requirements.

Chapter 10 describes a framework for adapting the ground distance of the EMD on fixed-binning histograms while preserving its metric properties. Given a metric EMD, mathematical constraints for the adaptation of the ground distance are developed together with a mapping of adaptation preferences to the space of feasible adaptations.

Chapter 11 proposes a relevance feedback process for the EMD on both fixed and adaptive-binning histograms. The method treats the ground distance of the EMD as a variable in an iterative optimization process. The evaluation shows that it can be well worth finding a suitable ground distance instead of relying on a default distance measure.
Chapter 10

Metric Adaptation of the Earth Mover’s Distance

While the efficiency of EMD-based multimedia retrieval has been explored in Part III of this work and in the related literature, the ground distances as the decisive factor for the effectiveness of the Earth Mover’s Distance has largely been treated as a given. Assuming that the ground distance is obvious from the application domain and static in nature is both overly simplifying the issue and sacrificing the flexibility of the EMD as one of its key advantages. In this chapter, a mathematical framework for adapting the ground distance of the EMD is proposed. The object of investigation of this chapter is the metric quality of the EMD for fixed-binning histograms. Given a metric EMD, how can the ground distance be adapted without losing the metric properties? Using the proposed mathematical framework, the ground distance of the EMD and thus the overall similarity model can be adapted to meet application or user requirements.

10.1 Introduction

Being based on a ground distance in the feature space, the Earth Mover’s Distance is a highly adaptable similarity measure. The ground distance determines how costly a transformation of one feature into another is. Changing the relative cost of transformation for pairs of feature values results in a changed overall
similarity measure as the Earth Mover’s Distance optimizes the overall cost of the transformation. Yet, the choice of the ground distance of the EMD has received comparatively little attention in the previous chapters and in much of the applicable literature. It is typically assumed that a suitable ground distance is apparent from the application domain or given by an expert user. However, since the feature space can be arbitrarily complex and contain dimensions from very different domains (e.g., spatial and spectral dimensions), it is prudent to consider the possibility of a ground distance that is not optimally suited for the retrieval task at hand. In this case changes to the relative transformation costs may result in a more effective retrieval process.

For some of the techniques that improve the efficiency of EMD-based retrieval, the ground distance is taken into account insofar as certain properties are assumed. Optimizations of the lower bound $LB_{iM}$ in [AWS06] and of the Lookup-Table lower bound in Chapter 9 require that the ground distance is metric (cf. page 14). Similarly, metric indexing as applied in [LBS06] requires a metric EMD. As shown in the appendix of [RTG00], a metric ground distance results in a metric EMD if the feature representations being compared share an equal total weight.

In this chapter, a mathematical framework for adapting the ground distance of the EMD on fixed-binning histograms of equal total weight is proposed. The adaptation is based on a number of preference statements by the user that demand relative changes to one or more of the transformation costs and ensures that the adapted EMD remains metric. In this way, the Earth Mover’s Distance can be adapted to potentially suit the application demands better than a default or static ground distance would. At the same time, efficiency improving techniques based on the metric property of the EMD remain applicable.

10.2 Mathematical Framework for Adapting a Metric EMD

Figure 10.1 gives an overview over the proposed framework for adapting a metric ground distance for the EMD. Given the according cost matrix $C$, limits $\alpha_{ij}$
10.2. Mathematical Framework for Adapting a Metric EMD

and \( \alpha_{ij}^+ \) for the adaptation are established in Section 10.2.1. Adaptations outside those limits result in a non-metric ground distance. Together with user preferences expressed by parameters \( \beta_{ij} \) and \( \lambda \) described in Section 10.2.2, a cost matrix \( C' \) that results in an adapted metric EMD is derived. Section 10.2.3 discusses some techniques to obtain preference parameters \( \beta_{ij} \) and \( \lambda \). Other techniques to determine preferences regarding relative transformation costs can be incorporated into the proposed framework as seen fit.

10.2.1 Limits of the Adaptability

As discussed in [RTG00], a metric ground distance results in a metric Earth Mover’s Distance (given feature representations of equal weight such as the fixed-binning feature histograms). It is easy to show that the converse holds, too.

Theorem 10.1 (Non-Metric ground distance implies non-metric EMD)

Given a non-metric ground distance \( gd \) for a feature space \( FS \), the Earth Mover’s Distance \( EMD_{gd} \) based on that ground distance is not metric.

Proof 10.1

If \( gd \) is non-metric, then by definition

1. \( \exists (r, t) \in FS \times FS : (r \neq t) \land (gd(r, t) = 0) \), or
2. \( \exists r \in FS : gd(r, r) \neq 0 \), or

3. \( \exists (r, t) \in FS \times FS : gd(r, t) \neq gd(t, r) \), or

4. \( \exists (r, t, u) \in FS \times FS \times FS : gd(r, u) > gd(r, t) + gd(t, u) \).

For signatures \( s^r = \{(r, 1)\} \), \( s^t = \{(t, 1)\} \), and \( s^u = \{(u, 1)\} \), it holds for case

1. \((s^r \neq s^t) \land (EMD_{gd}(s^r, s^t) = gd(r, t) = 0)\),

2. \( EMD_{gd}(s^r, s^t) = gd(r, r) \neq 0 \),

3. \( EMD_{gd}(s^r, s^t) = gd(r, t) \neq gd(t, r) = EMD_{gd}(s^t, s^r) \),

4. \( EMD_{gd}(s^r, s^u) = gd(r, u) > gd(r, t) + gd(t, u) = EMD_{gd}(s^r, s^t) + EMD_{gd}(s^t, s^u) \).

Thus, \( EMD_{gd} \) is not metric.

Since feature signatures in the proof are generalizations of fixed-binning feature histograms, the theorem holds for the latter as well, insofar as the non-metric properties of \( gd \) influence the cost matrix \( C \).

**Corollary 10.1**

The Earth Mover’s Distance \( EMD_C \) on fixed-binning histograms of dimensionality \( n \) with a cost matrix \( C \in \mathbb{R}^{n \times n} \) is metric iff

1. definiteness: \( \forall 1 \leq i, j \leq n : C[i, j] = 0 \iff i = j \),

2. symmetry: \( \forall 1 \leq i, j \leq n : C[i, j] = C[j, i] \), and

3. triangle inequality: \( \forall 1 \leq i, j, k \leq n : C[i, j] \leq C[i, k] + C[k, j] \).

If one of the properties is violated for the cost matrix \( C \), an example with an according violation of the metric properties of the EMD can easily be derived analogously to Proof 10.1. Since metric properties of the cost matrix as given by the corollary induce metric properties for the EMD on fixed-binning histograms, limits for adapting the ground distance of the EMD without sacrificing the metric property of the EMD can be determined in terms of \( C \).
The adaptation of an entry $C[i, j]$ within the framework proposed here will be executed via multiplication with a factor $\alpha_{ij}$. It is useful to define matrices that result from element-wise multiplication for this purpose.

**Definition 10.1 (Element-Wise Matrix Multiplication)**

Given matrices $A, B \in \mathbb{R}^{n \times m}$, the element-wise multiplication $A \bullet B$ is defined via

$$(A \bullet B)[i, j] = A[i, j] \cdot B[i, j].$$

**Definition 10.2 (Single-Element Matrix Multiplication)**

Given a matrix $A \in \mathbb{R}^{n \times m}$ and a factor $\alpha_{ij}$ associated with dimensions $i$ and $j$ of $A$, the matrix $(A \bullet \alpha_{ij})$ is defined as

$$(A \bullet \alpha_{ij})[i', j'] =
\begin{cases}
A[i', j'] \cdot \alpha_{ij} & \text{if } (i, j) = (i', j'), \\
A[i', j'] & \text{otherwise}.
\end{cases}$$

In the following theorems, necessary restrictions on factors $\alpha_{ij} = \alpha_{ji}$ are derived such that the EMD based on $((C \bullet \alpha_{ij}) \bullet \alpha_{ji})$ is metric if the EMD based on $C$ is metric.

The symmetry property holds by design due to $\alpha_{ij} = \alpha_{ji}$. The definiteness property holds for $\alpha_{ij} > 0$. The triangle inequality depends on relative values of the entries in $C$. Limits for $\alpha_{ij}$ are given in Theorem 10.2 and 10.3.

**Theorem 10.2 (Smallest upper bound)**

Given a cost matrix $C \in \mathbb{R}^{n \times n}$ of a metric Earth Mover’s Distance $EMD_C$ on fixed-binning feature histograms, the smallest upper bound $\alpha_{ij}^+ = \alpha_{ji}^+$ for factors $\alpha_{ij} = \alpha_{ji}$ ($i \neq j$) such that the triangle inequalities hold for $EMD_{C'}$ with $C' = ((C \bullet \alpha_{ij}) \bullet \alpha_{ji})$ is given by

$$\alpha_{ij}^+ = \min_{k \neq i, j} \{C[i, k] + C[k, j] \} / C[i, j].$$

**Proof 10.2**

The limit $\alpha_{ij}^+$ is derived such that at least one triangle inequality is fulfilled by an equality. Choosing $\alpha_{ij} > \alpha_{ij}^+$ directly leads to a violation of a triangle inequality.
Thus, $\alpha_{ij}^+$ is the smallest upper bound such that the triangle inequalities hold for $C^+ = ((C \circ \alpha_{ij}^+) \circ \alpha_{ij}^+)$. The adapted value $C^+[i,j]$ appears in three sets of triangle inequalities that have to be shown to still hold (and analogous sets for $C^+[j,i]$):

1. $\forall g : C^+[i,j] \leq C^+[i,g] + C^+[g,j]$
2. $\forall h : C^+[i,h] \leq C^+[i,j] + C^+[j,h]$
3. $\forall l : C^+[l,j] \leq C^+[l,i] + C^+[i,j]$

Due to the metric properties of $C$, according equations are known to hold for $\alpha_{ij} = 1$. For that reason, $\alpha_{ij}^+$ is known to be greater than or equal to 1.

Let $k^* = \arg \min_{k \neq i,j} \{C[i,k] + C[k,j]\}$.

1. For $g = k^*$, the inequality holds with equality:

$$C^+[i,j] = C[i,j] \cdot \alpha_{ij}^+ = C[i,k^*] + C[k^*, j] = C^+[i,k^*] + C^+[k^*, j].$$

For $g = i$, the inequality holds trivially:

$$C^+[i,g] + C^+[g,j] = C^+[i,i] + C^+[i,j] \geq C^+[i,j].$$

For $g = j$, the inequality holds analogously to $g = i$.

For all other $g$, the inequalities hold due to

$$C^+[i,j] = C[i,k^*] + C[k^*, j] \leq C[i,g] + C[g,j] = C^+[i,g] + C^+[g,j].$$

2. The inequality holds for $h = j$ due to case 1 with $g = j$ and it trivially holds for $h = i$.

For $h \neq i,j$, the changed value $C^+[i,j]$ appears only on the right hand side of the inequality and is greater or equal to the original value. Thus:

$$C^+[i,h] = C[i,h] \leq C[i,j] + C[j,h] = C[i,j] + C^+[j,h] \leq C^+[i,j] + C^+[j,h].$$

3. Analogous to case 2.
10.2. Mathematical Framework for Adapting a Metric EMD

Theorem 10.3 (Greatest lower bound)

Given a cost matrix \( C \in \mathbb{R}^{n \times n} \) of a metric Earth Mover's Distance \( \text{EMD}_C \) on fixed-binning feature histograms, the greatest lower bound \( \alpha_{ij} = \alpha_{ji} \) for factors \( \alpha_{ij} = \alpha_{ji} \) \((i \neq j)\) such that the triangle inequalities hold for \( \text{EMD}_{C'} \) with \( C' = ((C \cdot \alpha_{ij}) \cdot \alpha_{ji}) \)

is given by

\[
\alpha_{ij} = \max_{k \neq i,j} \{ |C[i,k] - C[k,j]| \} / C[i,j].
\]

Proof 10.3

The proof is based on the same reasoning as Proof 10.2 with \( \alpha_{ij} \leq 1 \), \( C^- = ((C \cdot \alpha_{ij}) \cdot \alpha_{ji}) \), and \( k^* = \arg \max_{k \neq i,j} \{ |C[i,k] - C[k,j]| \} \).

1. The first set of inequalities holds since the changed value appears on the left hand side of the inequalities and \( C^-[i,j] \leq C[i,j] \). It still holds if \( C^-[i,j] \)

and/or \( C^-[j,i] \) appear on the right hand side, too.

2. For \( h = i \), the inequality holds due to

\[
C^-[i,h] = C^-[i,i] = 0 \leq C^-[i,j] + C^-[j,i] = C^-[i,j] + C^-[j,h].
\]

For \( h = j \), the inequality holds due to

\[
C^-[i,h] = C^-[i,j] \leq C^-[i,j] + C^-[j,h].
\]

For all other \( h \), the inequalities hold due to

\[
C^-[i,j] + C^-[j,h] = |C[i,k^*] - C[k^*,j]| + C[j,h] \\
\geq |C[i,h] - C[h,j]| + C[j,h] \\
\geq C[i,h] - C[h,j] + C[j,h] = C[i,h] = C^-[i,h].
\]

3. Analogous to case 2.

For any value \( \alpha_{ij} \in [\alpha^-_{ij}, \alpha^+_{ij}] \cap (0, \infty) \), the adapted cost matrix induces a metric EMD. The influence of \( \alpha_{ij} \) on the EMD is illustrated in Figure 10.2 where the location of histograms with equal distance to a 3-dimensional query histogram
Figure 10.2: Iso-distance surfaces of the EMD in a three-dimensional feature representation space
10.2. Mathematical Framework for Adapting a Metric EMD

$q$ is displayed. The cost matrix of this example is set to

\[ C = \begin{pmatrix} 0 & 4 & 8 \\ 4 & 0 & 8 \\ 8 & 8 & 0 \end{pmatrix}. \]

Figure 10.2(a) and Figure 10.2(b) show the same data with the latter giving a perpendicular view onto the plane of equal-weight histograms. The iso-distance surface of the metric EMD based on $C$ has six vertices in this three-dimensional feature representation space. Each vertex is directly tied to an entry in the cost matrix. For example, the position of the vertex labeled $emd v_{31}$ can be calculated as

\[ emd v_{31} = q + (e_1 - e_3)/C[3,1] \]

with $e_i$ as the $i^{th}$ unit vector. The vertex $emd v_{31}$ is the position where a cost of 1.0 is generated by moving mass from dimension 3 of $q$ to dimension 1 of $emd v_{31}$. In Figure 10.2(c), the cost of moving mass from dimension 3 to dimension 1 is increased from $C[3,1] = 8$ to 12 via multiplication with $\alpha^-_{31} = 1.5$. The vertex moves closer towards $q$. Any greater value would result in a non-metric EMD and in a non-convex iso-distance surface as shown in Figure 10.2(d).

As Figure 10.2(e) indicates, it is also possible to decrease $C[3,1]$ from 8 to 4 without losing the convex property of the iso-distance surface. The EMD resulting from an adaptation according to $\alpha^-_{31}$ is again metric. Theorem 10.4 will show that such decreases in cost can instead be modeled as an increase of all other costs. Thus, the remainder of the chapter will treat adaptation values $\alpha_{ij} \in [1, \alpha^+_{ij}]$.

**Theorem 10.4 (Modeling a decrease as an increase)**

Given a cost matrix $C \in \mathbb{R}^{n \times n}$ of a metric Earth Mover’s Distance $EMD_C$ and factors $0 < \alpha_{ij} = \alpha = \alpha_{ji} < 1$, it holds that

\[ EMD_{C'}(s,t) = \alpha \cdot EMD_{C''}(s,t) \]

where $C' = ((C \bullet \alpha_{ij}) \bullet \alpha_{ji})$ results from $C$ via a decrease of costs $C[i,j]$ and $C[j,i]$
and $C''$ results from $C$ via an increase of all other costs by factor $\frac{1}{\alpha}$.

**Proof 10.4**

The cost matrix $C''$ that arises from an increase of all costs other than $C[i, j]$ and $C[j, i]$ can be stated as $C'' = \frac{1}{\alpha}(C \cdot \alpha_{ij}) \cdot \alpha_{ji}$. It only differs from $C'$ in the factor $\frac{1}{\alpha}$: $C'' = \frac{1}{\alpha}C'$. A cost matrix uniformly scaled with factor $\frac{1}{\alpha} > 1$ yields the same optimal flow $F^*$ for the EMD. Thus,

$$\alpha \cdot EMD_{C''}(s, t) = \alpha \cdot \sum_{i=1}^{n} \sum_{j=1}^{n} F^*[i, j]C''[i, j]$$

$$= \alpha \cdot \sum_{i=1}^{n} \sum_{j=1}^{n} F^*[i, j](-C')[i, j]$$

$$= \frac{\alpha}{\alpha} \cdot \sum_{i=1}^{n} \sum_{j=1}^{n} F^*[i, j]C'[i, j]$$

$$= EMD_{C'}(s, t).$$

Any adaptation of a metric EMD similarity measure via a symmetric decrease in values $C[i, j]$ and $C[j, i]$ can thus be modeled via an according increase of all other cost values. Figure 10.2(f) shows that the iso-distance surface for the EMD using matrix $C''$ equals the surface of Figure 10.2(e) except for scale. The vertices $emd_{v31}$ and $emd_{v13}$ remain in their original position (cf. Figure 10.2(b)) while all other vertices move closer to $q$. In many similarity search algorithms, the final scaling of the EMD values by factor $\alpha$ can either be skipped as only the relative distance values are important (e.g., ranking and nearest-neighbor queries) or can be incorporated in the similarity threshold (e.g., range queries).

### 10.2.2 Reconciling Adaptation Preferences with the Limits of Adaptability

After discussing the limits of the adaptability that stem from the original metric cost matrix in Section 10.2.1, the adaptation preferences are mapped to the interval of valid cost increases $[1, \alpha_{ij}^+]$. The specific way of determining adaptation preferences will be abstracted from within this section. A short treatment
of the topic is given in Section 10.2.3. For the remainder of this subsection, it is assumed that preferred adaption values $\beta_{ij} \in [0, 1]$ with $\beta_{ij} = \beta_{ji}$ and $\beta_{ii} = 0$ are given. A value of $\beta_{ij} = 0$ indicates a preference for no increase in cost while a value of $\beta_{ij} = 1$ indicates a preference for maximum possible cost increase for transformations from dimension $i$ to $j$. The aim is to find admissible modification values $\alpha_{ij} \in [1, \alpha_{ij}^+]$ that reflect the proportionate relation of the preferred adaptations $\beta_{ij}$ as much as possible. Ideally, $\beta_{ij} = h \cdot \beta_{ij'}$ results in an $h$ times stronger relative increase of $C[i, j]$ compared with $C[i', j']$.

As a first step, intermediate modification factors $\tilde{\alpha}_{ij} \in [1, \alpha_{ij}^+]$ that reflect the proportionate adaptation preferences are defined.

**Definition 10.3 (Intermediate modification factors)**

Given a cost matrix $C \in \mathbb{R}^{n \times n}$ of a metric Earth Mover's Distance $EMD_C$ and preferred cost increase values $\beta_{ij} \in [0, 1]$, the intermediate adaptation factors $\tilde{\alpha}_{ij}$ are defined by

$$\tilde{\alpha}_{ij} = \beta_{ij} \cdot (\alpha_{min}^+ - 1) + 1$$

with $\alpha_{min}^+ = \min\{\alpha_{ij'}^+ | (1 \leq i', j' \leq n) \wedge (i' \neq j') \wedge (\beta_{ij'} > 0)\}$ and $\alpha_{ij'}^+$ according to Theorem 10.2.

Since $\alpha_{min}^+$ is the smallest of all relevant upper bounds for the adaptation and since $\beta_{ij} \leq 1$, all intermediate adaptation values are within admissible limits. Furthermore, the definition of the upper bounds implicates that an increase of one cost entry in the cost matrix does not decrease the limit of possible increases for any other cost entry. Thus, all adaptations $\tilde{\alpha}_{ij}$ can be executed together without jeopardizing the metric properties. With $\tilde{\alpha} = [\tilde{\alpha}_{ij}]$ as the matrix of intermediate adaptation values, the intermediate adapted matrix $\tilde{C}$ is given by $\tilde{C} = C \cdot \tilde{A}$. If there is a preference to increase a cost value that may on its own not be increased without losing the metric property, the preferences can only partially be heeded in this manner. If $\alpha_{min}^+ = 1$, the offending preference may either be disregarded or (partially) be regarded in a later adaptation.

Factors $\tilde{\alpha}_{ij}$ from Definition 10.3 result in a metric adaptation of the cost matrix within a minimal admissible interval. It is not guaranteed that the largest admissible increase that respects the relative preferences has been found by $\tilde{\alpha}_{ij}$.
By adding a single additional parameter $\lambda \in [0, 1]$, it is possible to control the overall adaptation magnitude. Together with this parameter the adaptation factors $\alpha_{ij}$ can be defined as follows.

**Definition 10.4 (Adaptation factors)**

Given a cost matrix $C \in \mathbb{R}^{n \times n}$ of a metric Earth Mover’s Distance $EMD_C$, preferred cost increase values $\beta_{ij} \in [0, 1]$, and an overall magnitude of adaptation $\lambda$, adaptation factors $\alpha_{ij}$ are defined as

$$\alpha_{ij} = \lambda \cdot \rho \cdot (\tilde{\alpha}_{ij} - 1) + 1$$

where $\rho$ is to be chosen such that $\alpha_{ij} \leq \alpha_{ij}^+ \text{ for all } 1 \leq i, j \leq n$ and $\alpha_{ij} = \alpha_{ij}^+$ for at least one pair $(i, j)$ for $\lambda = 1$.

The missing factor $\rho$ that ensures that a maximum adaptation magnitude is reached for $\lambda = 1$ can be derived from the smallest increase of the intermediate factors (on a percentage basis).

**Theorem 10.5**

For $\alpha_{ij}$ according to Definition 10.4 with $\lambda = 1$ and

$$\rho = \min \left\{ \frac{\alpha_{i'j'}^+ - 1}{\tilde{\alpha}_{i'j'} - 1} \middle| (1 \leq i', j' \leq n) \land (i' \neq j') \land (\beta_{i'j'} > 0) \right\}$$

it holds that

$$\exists (i^*, j^*) : \left( \alpha_{i^*j^*} = \alpha_{i^*j^*}^+ \land \forall (i', j') \neq (i^*, j^*) : \alpha_{i'j'} \leq \alpha_{i'j'}^+ \right)$$

**Proof 10.5**

For $(i^*, j^*)$ as a pair of indices for which $\rho$ is minimal according to Theorem 10.5
it holds:

\[ \alpha_{i'j'} = \lambda \cdot \rho \cdot (\tilde{\alpha}_{i'j'} - 1) + 1 \]

For \((i', j') \neq (i^*, j^*)\) with \(i' \neq j'\) and \(\beta_{i'j'} > 0\) it holds:

\[ \alpha_{i'j'} = \lambda \cdot \rho \cdot (\tilde{\alpha}_{i'j'} - 1) + 1 \]

\[ \leq 1 \cdot \rho \cdot (\tilde{\alpha}_{i'j'} - 1) + 1 \]

For \(i' = j'\) and for \(\beta_{i'j'} = 0\), it holds that \(\alpha_{i'j'} = 1\) and thus the triangle inequalities remain unaffected.

Definition 10.4 enables the metric adaptation \(C' = C \cdot [\alpha_{ij}]\) of the Earth Mover’s Distance on fixed-binning histograms based on cost increase preferences \(\beta_{ij}\). A few exemplary choices for deriving preferences \(\beta_{ij}\) from interaction with the user are discussed next.

### 10.2.3 Determination of Adaptation Preferences

The proposed metric adaptation of the Earth Mover’s Distance modifies the cost matrix of the EMD based on adaptation preferences \(\beta_{ij}\). The mathematical framework is independent from the manner in which preferences \(\beta_{ij}\) are determined. Some options for deriving according values heuristically are discussed in the following paragraphs.

If elements from the feature space in which the ground distance of the EMD is calculated can be illustrated appropriately, a user may be asked to assign scores for the dissimilarity of pairs of histogram bin representatives. For ex-
ample, in the case of a color space as the feature space, two bin colors can be shown to the user. An assessment on a scale from “identical/very similar” to “very dissimilar” can then be mapped to appropriate values for $\beta_{ij}$ if the degree of dissimilarity is not sufficiently reflected by the default cost matrix. It may however be unreasonable to expect the user to be able to give consistent scores for multiple feature pairs. In that case, it is possible to simplify the decision for the user by presenting three features for which only the relative dissimilarity has to be decided upon. E.g., given a turquoise bin representative, the user could be asked to decide if it is to be treated as “more green” or “more blue” by asking for the more similar color of the two. In this manner, mismatches of the default ground distance and the perception of the user can be determined and turned into preference values $\beta_{ij}$.

Instead of asking for assessments in a conceivably abstract feature space, techniques similar to those found in Relevance Feedback systems (cf. Chapter 11) can be used to derive information regarding the perception of similarity in the feature space based on feedback regarding the similarity of objects in a database. Given sets of similar objects as determined by the user, it may be possible to draw conclusions regarding the relative perception of similarity in the feature space from statistical information gathered for the feature representation of those objects. A related method is proposed in Section 11.2.3 of the next chapter. Together with such data-based techniques the framework may be used to form an EMD-specific Distance Metric Learning [YJ06] method.

10.3 Summary

This chapter proposed a mathematical framework for adapting a metric Earth Mover’s Distance on fixed-binning histograms. Given a method to derive adaptation preferences, it produces an adapted Earth Mover’s Distance that is again metric. This allows for query processing algorithms and optimizations that are based on metric properties to be utilized.

Guided by insight gained through analyzing the iso-distance surface of the EMD, limitations of metric adaptations are derived based on the initial cost
matrix of the EMD. Next, a method for mapping multiple given adaptation preferences to adaptation factors was proposed such that the resulting EMD both conforms to the limitations of metric adaptation and proportionally reflects the adaptation preferences. Finally, several options for deriving adaptation preferences were pointed out.
Chapter 11

Exploring Multimedia Databases via Relevance Feedback

In this chapter, a technique to adapt the EMD to feedback from the user of a similarity search system is proposed. It is applicable to both fixed-binning histograms and adaptive-binning histograms (signatures). In contrast to the last chapter, the metric property is sacrificed in return for an improved flexibility that can be exploited to increase the retrieval effectiveness of the similarity search system. While not the focus of this chapter, some methods that improve the efficiency of the approach can still be applied without relying on metric properties of the EMD.

11.1 Introduction

Two common challenges of distance-based similarity search are the formulation of the query and the definition of a suitable distance measure. Query objects in similarity search based on the query-by-example paradigm by definition can be no more than imprecise descriptions of what users are looking for in the database. Additionally, even for high-quality similarity models, the distance measure can only be an approximation of the users' notion of similarity as said notion may be different from application to application, from user to user, and ultimately from query to query.
Distance-based Relevance Feedback (RF) approaches [Roc71, ISF98, OBM04, FdSTGF08] address these issues on the basis of relevance information gathered from the user. They aim to increasingly reflect the user’s notion of similarity and return a larger amount of relevant objects from the database. Returning more relevant objects in earlier iterations turns effectiveness of the similarity model into efficiency for the user in this scenario.

An extension of the traditional feedback loop (cf. Fig. 11.1) is proposed in this chapter that takes an adaptable similarity measure (i.e., the EMD) and improves the quality of the search result through back-testing based on prior feedback. To this end, the similarity adaptation is formulated as an optimization problem that minimizes discrepancies between relevance information and the distance measure. Two statistics-based heuristics for the adaption of the EMD to feedback information are introduced as initializations to the optimization process – one for fixed-binning histograms and one for adaptive-binning histograms. Experiments on real world databases show that significantly more relevant objects are returned in fewer iterations resulting in a faster exploration of the database than existing EMD-based query movement techniques allow for.

11.1.1 Related Work

The distance-based Relevance Feedback framework allows for the utilization of database techniques such as spatial access methods. As an early example of
Relevance Feedback for the vector space model, [Roc71] proposed the reformulation of a query by moving the vector that represents the query towards known relevant objects and away from known non-relevant objects. Systems such as MindReader [ISF98] and MARS [OBM04] complement the query movement by determining an adapted distance function based on (co-)variance information present in the feedback.

In the text retrieval domain, an algorithm for basing the adaptation of the query on an iterative optimization process that takes the user’s feedback into account was proposed in [BS95]. For their text retrieval system, a query consists of a number of terms and according weights that can be updated by the optimization process. Another optimization-based RF approach that adapts a similarity measure was recently presented in [FdSTGF08]. It uses genetic programming to optimize the arithmetic combination of a number of general similarity functions. Inspired by the Simulated Annealing (SA) optimization technique, [CFPF03, CPFGF05] propose a system where each iteration of a feedback process is a single iteration in a search process that becomes more and more focused over time akin to the idea of SA. Some image retrieval systems based on RF break with the convention that each object in the database has to be described by a single vector. Region-based RF [JLZ+03, LH08] exploits the position, size, shape, and/or feature distribution of connected regions within images. Recently, Li et al. [LH08] proposed representing objects as graphs with nodes that represent image regions and edges between nodes of neighboring regions. The nodes are annotated with feature information for the corresponding regions. Their RF process uses two optimization steps – one for matching graphs and the other for updating the query graph. While the features stored in the nodes of the query graph are updated using relevance information, the structure of the query graph itself remains unchanged, which limits the flexibility of the approach.

In the limited number of Relevance Feedback approaches where the EMD is utilized as a similarity measure, the EMD itself is not adapted to the user feedback but instead the approaches rely on its good default retrieval effectiveness. In [RT01], a Relevance Feedback approach termed *Query-by-Refinement* is proposed that combines feature representations from relevant objects to a rep-
representation of a new virtual query object. Thus, *Query-by-Refinement* performs query adaptation only but the EMD itself remains unchanged. In [JLZ+03], the EMD is used as a kernel function for an SVM-based Relevance Feedback system. The query results are iteratively improved by re-weighting the query signature and training an SVM classifier according to user feedback. The feedback is not used to adapt the underlying EMD.

The flexibility of the EMD with regards to its ground distance will be exploited in the following sections by adapting the ground distance based on feedback from the user.

### 11.2 Relevance Feedback with the Earth Mover’s Distance

The main Relevance Feedback process for the Earth Mover’s Distance is formally introduced in section 11.2.1. Section 11.2.2 describes the proposed optimization-based extension of the feedback process that adapts a similarity measure such that the discrepancy between all user-given feedback and the distance measure is minimized. It is not strictly specific to the Earth Mover’s Distance and can be adopted in conjunction with other flexible similarity measures. Sections 11.2.3 and 11.2.4 detail proposals for EMD-based Relevance Feedback on feature histograms and on signatures. A statistics-based heuristic for adapting the Earth Mover’s Distance is proposed in each section. The heuristics can be used on their own or serve as initial solutions for the optimization-based extension from Section 11.2.2.

#### 11.2.1 The Feedback Process

Algorithm 11.1 lists pseudo-code for the Relevance Feedback process that is the basis for the remainder of the chapter. It acts as a framework for all following techniques in that it takes as input parameters two procedures for adapting the query representation and the ground distance of the EMD. Differing implementations will be defined in later sections leading to differing relevance feedback
11.2. Relevance Feedback with the Earth Mover’s Distance

Algorithm 11.1: feedbackLoop(DB, q, gd, k, computeQuery, computeGD)

1  iter = 1;
2  results = nnQ(q, k, \(EMD_{gd}\), DB);              // see page 22
3  while (getUserSatisfied(results) == false) do
4    feedback[iter].rel = getRelUserFeedback(results);
5    feedback[iter].nonrel = results - feedback[iter].rel;
6    q = computeQuery(feedback, iter);
7    gd = computeGD(q, feedback, iter);
8    results = nnQ(q, k, \(EMD_{gd}\), DB);
9    iter = iter + 1;
10   endwhile

systems.

Given an initial query object \(q\) from the user and a default ground distance \(gd\) such as \(d_{L^2}\), a Relevance Feedback session starts by retrieving a \(k\)-subset of the objects in the database \(DB\) via a nearest-neighbor query using the Earth Mover’s Distance with the default ground distance.

After returning the subset to the user (e.g., in a graphical user interface), the feedback process halts and waits for feedback from the user. Unless the user is already satisfied with the results or otherwise decides to terminate the feedback process, the main feedback loop is entered and repeated until the aforementioned condition holds.

Within the feedback loop, the user is asked to let the system know which of the objects in the subset are to be considered relevant. All other objects are considered non-relevant. Using this relevancy information and possibly also relevancy information from past iterations, the main task of the system is to find a new query object representation and a new ground distance for the Earth Mover’s Distance that better fit the user’s requirements. These two steps are defined by the implementation of the input parameters \(\text{computeQuery}\) and \(\text{computeGD}\) where the focus of this chapter lies in finding suitable feedback-based ground distances. Lastly, a new \(k\)-subset of objects similar to the new computed query \(q\) as defined by the Earth Mover’s Distance with the updated ground distance is retrieved and the loop starts anew.

The next section describes how the Relevance Feedback loop from Algo-
Algorithm 11.1 can be extended with an optimization-based step that improves on heuristically computed ground distances.

### 11.2.2 Optimization of the Similarity Measure

While heuristics for computeGD such as the ones presented in sections 11.2.3 and 11.2.4 are able to produce ground distances for the EMD that enable the retrieval of more relevant database objects than is possible without adapting the ground distance, they do not necessarily reflect the user’s feedback in an optimal way. Instead of returning the results as determined by the EMD based on the heuristic ground distance, the proposed extension to the feedback loop depicted in Figure 11.2 first tests if the ground distance can be improved upon. The adaptation of the ground distance by the extension is phrased as an optimization problem that ties the ground distance as the optimization variable to the consistency of the EMD-based similarity model with the user feedback as the optimization criterion. The approach is conceptually related to [BS95], where a greedy optimization was chosen to find an improved query representation via changing term weights in text retrieval. Unlike [BS95], the similarity measure itself is optimized here. This concept is not exclusively tied to the EMD and can be transferred to other flexible distance measures such as quadratic form distances where the orientation and the extent of the ellipsoid-shaped iso-distance surface would be subject to the optimization.
Adaptation as an Optimization Problem

The central building block of phrasing the adaptation of a similarity measure as an optimization problem is the definition of a suitable optimization criterion. Obviously, the optimization process cannot know if including an arbitrary object from the database in the result set will increase the number of relevant objects returned to the user. It is not possible to adapt the similarity measure, compute a new $k$-nearest-neighbors set and check if the adaptation was favorable in terms of, for instance, recall/precision since the optimization process does not have access to the user to determine the quality of the result. Otherwise the extension would merely be another feedback loop.

However, it is possible to test if the adaptation resulted in a similarity measure that is consistent with the user’s past feedback. To this end, the objects from all previous result sets are ranked according to the adapted similarity measure. A good measure results in a ranking of the feedback where the objects already identified as relevant appear before those not identified as relevant. An unsuitable measure has all non-relevant objects appear before the relevant ones.

To automatically decide how beneficial a given ranking of the feedback is, a quality measure that reduces the ranking to a single value is required. One example of such a quality measure is given in the following definition. Other quality measures for rankings from the literature can also be used.

**Definition 11.1 (Average Precision at Relevant Positions)**

Given a database $DB$ of objects, relevant objects $\mathbb{R} \subseteq DB$, and a ranking function $\text{rank}: DB \rightarrow \mathbb{N}_1$, the average precision of the ranking is defined as

$$\text{avgPrecision}(\text{rank}, \mathbb{R}) = \frac{1}{|\mathbb{R}|} \sum_{\hat{o} \in \mathbb{R}} \frac{|\{\hat{o} \in \mathbb{R} | \text{rank}(\hat{o}) \leq \text{rank}(o)\}|}{\text{rank}(o)}$$

Intuitively, a higher number of relevant objects appearing at the front of the ranking results in a higher avgPrecision-value. A ranking with all relevant objects at the front of the ranking has the optimal value of 1. The following table gives an example for four different rankings where relevant objects are denoted by 1 and all other objects are denoted by 0.
Using a measure such as avgPrecision for judging the quality of a ranking, it is possible to formulate the similarity measure adaptation as follows.

**Definition 11.2 (Optimal Relevancy-Consistent Similarity Measure)**

Given query \( q \), database \( DB \), and relevant objects \( \mathcal{R} \subseteq DB \), the optimal relevancy-consistent similarity measure \( \text{Sim}^* \) with respect to avgPrecision from Definition 11.1 is

\[
\text{Sim}^* = \arg \max_{\text{Sim}} \{ \text{avgPrecision}(\text{rank}_{\text{Sim}}, \mathcal{R}) \}
\]

where \( \text{rank}_{\text{Sim}} \) determines the ranking of objects in \( DB \) by descending similarity to \( q \) according to similarity measure \( \text{Sim} \).

During the Relevance Feedback process, the quality measure is evaluated on the subset of the database for which the user has already given relevancy information to the system. By optimizing the quality measure on the objects fed back to the system, the final similarity measure will be one that in the best case separates known relevant from known non-relevant objects. Objects for which the user has not given any feedback are not taken into account as the system cannot know if they are to be considered relevant or not without asking the user.

**The Optimization Algorithm**

The goal of the optimization algorithm in this section is to find \( \text{Sim}^* \) of Definition 11.2. The distance and ranking computations necessary to calculate the value of the optimization criterion make the influence that the parameters of the similarity measure have on this optimization criterion all but trivial and an analytical optimization infeasible. This is especially true in the case of the Earth Mover's Distance, where the parameter that defines the EMD-based similarity measure is the ground distance function and where the computation of

<table>
<thead>
<tr>
<th>ranking</th>
<th>avgPrecision</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 1 1 0 0 0</td>
<td>1.000</td>
</tr>
<tr>
<td>1 1 0 1 0 0</td>
<td>0.950</td>
</tr>
<tr>
<td>0 0 1 1 1 0</td>
<td>0.525</td>
</tr>
<tr>
<td>0 0 0 1 1 1</td>
<td>0.430</td>
</tr>
</tbody>
</table>
11.2. Relevance Feedback with the Earth Mover’s Distance

Algorithm 11.2: optimize(C, q, feedback)

1  extern maxIter, coolFactor, $T^0$;
2  optIter = 0; $T = T^0$;
3  avgpOld = avgpBest = avgpNew = avgPrecision(C, q, feedback);
4  Cbest = C;
5  while (optIter < maxIter) do
6      $C' = \text{modify}(C)$; // move in solution space
7      avgpNew = avgPrecision($C'$, q, feedback); // evaluate solution
8      if (avgpNew $\geq$ avgpOld) || (rand() $<$ $\exp((\text{avgpNew} - \text{avgpOld})/T)) then
9          $C = C'$; avgpOld = avgpNew; // adopt new solution
10         if (avgpNew $\geq$ avgpBest) then
11             Cbest = C'; avgpBest = avgpNew; // remember best solution
12         endif
13      endif
14      $T = T * \text{coolingFactor}$; // turn search less erratic
15      optIter = optIter + 1;
16  endwhile
17  return Cbest;

the optimization criterion requires the iterative optimization of multiple transformation problems according to Definition 7.2. However, there is a subclass of optimization algorithms that is well-suited for finding an approximate solution this situation.

Algorithm 11.2 is an instance of the family of probabilistic optimization heuristics referred to as Simulated Annealing algorithms [KGV83]. The main idea is to start the optimization at a given point in the solution space (i.e., parameter collection $C$ that defines the similarity measure) and randomly navigate through the solution space (via procedure modify) while evaluating the optimization criterion ($\text{avgPrecision}$) on the way. Unlike greedy algorithms, worse solutions can temporarily be adopted with a certain probability in order to overcome local optimums. The worse a solution is, the less likely is its adoption. In addition, the probability also decreases over time as the so-called temperature of the optimization process ($T$) tends towards zero. The optimization terminates after a set number of iterations ($\text{maxIter}$). In this fashion, the algorithm first moves rather erratically while looking for regions with good solutions. As it
progresses, it slowly converges towards a greedy algorithm. At the end, the best solution found during the optimization process is returned. Figure 11.3 shows four examples of the progress. The average precision of the currently adopted solution ($\text{avgpOld}$) may at first decrease in order to get out of local maximums. In later iterations, worse solutions are rarely adopted.

Choices of the annealing parameters $\text{coolFactor}$, $\text{maxIter}$, and $T^0$ that proved suitable for the EMD Relevance Feedback process are listed the evaluation section. The $\text{modify}$ procedure depends on the variability of the similarity measure. EMD-specific definitions are given in sections 11.2.3 and 11.2.4. The extended feedback framework presented in this section can be utilized to optimize other adaptable distance functions by defining a suitable modification procedure.

### 11.2.3 EMD-based Relevance Feedback on Histograms

Unlike feature signatures, fixed-binning histograms\(^*\) share a common partitioning of the feature space. As a consequence, it suffices to partially define the ground distance $gd$ between those partitions. Adapting the Earth Mover’s Dis-

\(^*\)Referred to as histograms in the rest of the chapter.
11.2. Relevance Feedback with the Earth Mover’s Distance

Algorithm 11.3: computeGD-QM(q, feedback, iter)

1. return $d_{L_2}$;

tance is reduced to adapting a finite set of distance values that are specified in a cost matrix $C \in \mathbb{R}^{n \times n}$ where $n$ is the dimensionality of the histograms. Possibilities and limits of adapting a metric cost matrix $C$ based on directly expressed user preferences were described in Chapter 10. This section will develop an automatic derivation of a cost matrix based on user feedback.

In the following two subsections, the adaptation of the query representation $q$ and a statistics-based heuristic for adapting the ground distance of the EMD based on the user feedback are described. The subsection hereafter shows how the optimization algorithm from section 11.2.2 ties in with the partially defined ground distance for the case of fixed-binning histograms.

Query Adaptation

Since the focus of this chapter is on adapting the similarity measure, a simple query reformulation that has been used in several distance-based Relevance Feedback systems is adopted.

Definition 11.3 (Query adaptation)

Given a set $\mathcal{R}_i$ of relevant fixed-binning feature histograms in iteration $i$ of the relevance feedback process, the new query representation $q_{i+1}$ for the next iteration is defined as

$$q_{i+1} = \frac{1}{|\mathcal{R}_i|} \sum_{o \in \mathcal{R}_i} o$$

The new query $q_{i+1}$ is shifted to the center of relevant feature histograms in the feature representation space. Passing an according algorithm and the procedure computeGD-QM listed in Algorithm 11.3 to the feedback algorithm from page 173 results in a baseline Relevance Feedback process that uses the Euclidean distance as the fixed ground distance for the EMD and moves the query to the center of relevant objects after each iteration.
Algorithm 11.4: computeGD-Histogram-Heur(q, feedback, iter)

1. if |feedback[iter].rel| ≤ 1 then return $d_{L2}$;
2. $C^{\text{Heur}} = \text{computeGD-Histogram-Heur-Matrix}(q, \text{feedback}, \text{iter})$;
3. return new CostMatrixDistance($C^{\text{Heur}}$);

Algorithm 11.5: computeGD-Histogram-Heur-Matrix(q, feedback, iter)

1. $\Sigma^* = \text{minFixedInverse}(\text{covmatrix(feedback[iter].rel)})$;
2. for $i = 1$ to $|\Sigma^*.\text{rows}|$ do
3.   for $j = 1$ to $|\Sigma^*.\text{cols}|$ do
4.     $C^{\text{Heur}}[i,j] = \sqrt{\Sigma^*[i,i] + \Sigma^*[j,j] - 2\Sigma^*[i,j]}$;
5.   endfor
6. endfor
7. return $C^{\text{Heur}}$;

Heuristic for Adapting the Ground Distance

The ground distance of the Earth Mover’s Distance on feature histograms describes transformation costs between individual feature space partitions shared by all feature histograms. The ground distance can be partially but sufficiently specified by a cost matrix $C$. Here, the transformation costs of the Earth Mover’s Distance shall be defined on the basis of the dissimilarity between the feature space partitions common to all relevant objects $\mathcal{R}_i$. The dissimilarity between feature space partitions is estimated as the dissimilarity between histograms of objects that have all features concentrated in a single partition. Thus, the aim is to define cost matrix entry $C[i,j]$ as

$$C[i,j] \approx \text{dissimilarity}(e_i, e_j)$$

where $e_i$ and $e_j$ are unit vectors with the same dimensionality as the feature histograms. The dissimilarity between those histograms is to be derived from the user feedback.

Based on the feature distribution described by the histograms of the relevant objects, the Mahalanobis distance defines a dissimilarity measure that fits the requirements. The Mahalanobis distance for a given feature distribution is
11.2. Relevance Feedback with the Earth Mover’s Distance

computed using its inverse covariance matrix $\Sigma^{-1}$:

$$d_{\text{Man-}\Sigma}(s, t) = \sqrt{(s - t) \cdot \Sigma^{-1} \cdot (s - t)^T}$$

Using the Mahalanobis distance on the histograms $e_i$ and $e_j$ to determine a dissimilarity between feature space partitions $i$ and $j$, appropriate transformation costs $C[i,j]$ can be computed.

$$C[i,j] = d_{\text{Man-}\Sigma}(e_i,e_j)$$

$$= \sqrt{(e_i - e_j) \cdot \Sigma^{-1} \cdot (e_i - e_j)^T}$$

$$= \sqrt{\Sigma^{-1}[i,i] + \Sigma^{-1}[j,j] - 2 \cdot \Sigma^{-1}[i,j]}$$

The inverse covariance matrix $\Sigma^{-1}$ is derived from the feature histograms determined as relevant in the current feedback iteration by the user. Directions in the feature representation space with a large variability in the feedback result in a small transformation cost while directions with a compact distribution of values result in a high transformation cost. The intuition of the heuristic is that compact feedback represents a preference by the user to stay in the according area of the feature representation space.

If the number of relevant feature histograms is small, the inverse covariance matrix may not be defined since some of the eigenvalues of the covariance matrix equal zero. Intuitively, this means that there is not enough information in the feedback to estimate dissimilarity in all directions of the feature representation space. To overcome this obstacle, eigenvalues equal to zero can be replaced with the smallest eigenvalue greater than zero during the matrix inversion process as suggested in [YX03]. This effectively sets the dissimilarity of directions without variability to a high value without making them infinitely dissimilar.

The procedure $\text{computeGD-Histogram-Heur}$ of Algorithm 11.4 computes the proposed heuristic adaptation of the Earth Mover’s Distance on fixed-binning histograms with the help of Algorithm 11.5, which reflects equation (11.1) with accommodations for eigenvalues of zero. The result is a distance function that is defined via $C^{\text{Heur}} \in \mathbb{R}^{n \times n}$ for the $n$ feature partition representatives of the histograms.
Algorithm 11.6: computeGD-Histogram-Opt(q, feedback, iter)

1. if \( |\text{feedback[iter].rel}| \leq 1 \) then return \( d_{L_2} \);
2. \( C^{\text{Heur}} = \text{computeGD-Histogram-Heur-Matrix}(q, \text{feedback}, \text{iter}) \);
3. \( C^{\text{Opt}} = \text{optimize}(C^{\text{Heur}}, q, \text{feedback}) \);
4. return new CostMatrixDistance\((C^{\text{Opt}})\);

Similarity Optimization

To apply the optimization-based similarity adaptation from section 11.2.2 to the EMD on fixed-binning feature histograms, two things are required:

1. a mapping from the ground distance of the EMD to the parameter collection that the optimization algorithm treats as possible solutions and
2. a method that randomly modifies the parameter collection to navigate through the solution space.

The parameter that defines the EMD is its ground distance in the form of a cost matrix. As an initial solution, the heuristically determined cost matrix \( C^{\text{Heur}} \) can be passed to the optimization algorithm 11.2.

Algorithm 11.7 modifies a cost matrix according to the externally set parameter \( \text{modRowWeight} \) that determines the modification magnitude. Each row in the cost matrix reflects the transformation costs from one histogram partition of the query object to the histogram partitions of objects in the database. For each row, \( \text{modRowWeight} \) determines how much the row is to be changed. The total size of the matrix is \( n \times n \) where \( n \) is the dimensionality of the histograms.

To update the row, its entries are randomly partitioned into two sets of equal cardinality \( \pm 1 \). The entries of the first partition are increased by \( \text{modRowWeight} \) percent of the sum of row costs while the entries of the second partition are decreased accordingly. The roles of the two entry partitions are switched if a decrease of the second partition would lead to negative costs. One of the two modifications is always possible with \( \text{modRowWeight} \) from \([0, 0.5)\). Lastly, the new cost matrix is returned to the optimization algorithm for evaluation via the average precision measure and possibly further iterative modification.
11.2. Relevance Feedback with the Earth Mover’s Distance

**Algorithm 11.7: modify(C)**

```plaintext
1. extern modRowWeight;
2. C’ = C;
3. for row ∈ C’.rows do
   4. delta = |row| * modRowWeight;
   5. (entries 1, entries2) = randomPartitioning(row);
   6. sum1 = |entries1|; sum2 = |entries2|;
   7. if sum2 ≤ delta then delta = -delta;
   8. for entry1 ∈ entries1 do entry1 = entry1 * (sum1 + delta) / sum1;
   9. for entry2 ∈ entries2 do entry2 = entry2 * (sum2 - delta) / sum2;
4. endfor
11. return C’;
```

11.2.4 EMD-based Relevance Feedback on Signatures

As discussed in Section 1.1, adaptive-binning histograms or feature signatures of the form \( s = \{(r_1, w_1), \ldots, (r_n, w_n)\} \) are more flexible in representing a feature distribution than fixed-binning histograms. In this section, the flexibility of the EMD on signatures is exploited for Relevance Feedback.

The structure of this section follows the same outline as the section on EMD adaptation for fixed-binning histograms. After a short description of the query adaptation, a statistics-based heuristic for the derivation of a ground distance from user feedback is proposed and then optimized via the extension to the feedback loop from Section 11.2.2.

**Query Adaptation**

There exist some methods for combining several feature signatures into a new signature in the literature [RT01, JLZ+03]. They are based on region reweighting [JLZ+03] and clustering of partition representatives [RT01]. Since the focus here is on adapting the similarity measure, the approach from [RT01] with a simple k-means algorithm was implemented for the experiments of Section 11.3. The main idea is to collect all signature components from all relevant signatures, cluster their representatives, and set the new weights to the median weight of the components in the clusters. Clusters with contributions from less than half of
**Algorithm 11.8**: computeGD-Signature-Heur(q, feedback, iter)

1. if $|\text{feedback}[\text{iter}].\text{rel}| \leq T$ then return $d_{L_2}$;
2. $C^{\text{Heur}} = \text{computeGD-Signature-Heur-Matrix}(q, \text{feedback}, \text{iter})$;
3. return new Distance($C^{\text{Heur}}$);

the relevant signatures are removed and a normalization of the weights ensures that the total weight is not greater than 1. Together with a fixed Euclidean ground distance produced by computeGD-QM from page 179, this reflects the Query-by-Refinement Relevance Feedback algorithm from [RT01] and serves as a baseline for the similarity adaptation algorithms on feature signatures proposed in the following subsections.

**Heuristic for Adapting the Ground Distance**

EMD-based Relevance Feedback on signatures requires a new adaptation heuristic. Since the partitioning representatives are different for every object in the database, their signatures do not live in a simple $n$-dimensional space for which statistical information could be captured in the same way as in Section 11.2.3. In addition, a cost matrix does not suffice in the signature case. Instead, a ground distance $gd: FS \times FS \to \mathbb{R}$ has to be defined for the feature space $FS$ in which the partitioning representatives of the signature components are situated.

The proposal in this section defines a ground distance based on the variance information of the partitioning representatives derived from feedback information. The key to defining a suitable ground distance is the observation that the ground distance does in fact not have to be defined for each pair of points in $FS$ but only for $\left(\bigcup_{(r^q, w^q) \in q} r^q\right) \times FS$. For each iteration of the feedback process, the similarity search process computes the Earth Mover’s Distance from a single object $q$ to objects in the database. Thus, a ground distance that defines distance values from the partitioning representatives of $q$ to points in $FS$ fully suffices.

The heuristic for signatures utilizes the local variance information of the feedback around the query representatives to partially define the ground distance. If representatives from the feedback are within a compact region in a dimension of the feature space around a query representative, the heuristic as-
11.2. Relevance Feedback with the Earth Mover’s Distance

Algorithm 11.9: computeGD-Signature-Heur-Matrix(q, feedback, iter)

// assign feedback representatives to query representatives
for s ∈ feedback[iter].rel do
  for (r^s, w^s) ∈ s do
    i^* = arg min_i \{d_{L2}(r^s, r^q_i) | (r^q_i, w^q_i) ∈ q\};
    Subset[i^*] = Subset[i^*] ∪ \{r^s\}; // remember that r^s is closest to r^q_i
  endfor
endfor
// for query representative, compute variance of assigned feedback
for i = 1 to |q| do
  varianceVector = variance(Subset[i]);
  // for each representative dimension
  for j = 1 to ~n do
    C^Heur[i, j] = 1 / varianceVector[j];
    // normalize row to sum of 1.0
  endfor
endfor
return C^Heur;

signs a high cost to transports out of this region. If the user picked relevant objects with a large variance around a query representative, the according cost is chosen to be low.

The procedure computeGD-Signature-Heur in Algorithm 11.8 takes all signatures of objects deemed relevant by the user in the current feedback iteration and assigns all their representatives to the closest query representative via Algorithm 11.9. For each query representative r^q_i, Subset[i] collects the representatives from the feedback that are closer to r^q_i than to any r^q_j (i ≠ j). The dimension-wise variance information of Subset[i] is stored in the i^{th} row of a matrix C^Heur, where C^Heur[i, j] is normalized inverse variance of Subset[i] in the j^{th} dimension of the feature space FS. The matrix C^Heur is of size |q| × ~n where |q| is the number of representatives in the query signature and ~n is the dimensionality of the feature space FS.

Using the variance information in C^Heur, the ground distance between a query representative r^q_i and an arbitrary other representative r ∈ FS is com-
computed as a weighted Euclidean distance

\[
gd(r^q_i, r) = \sqrt{(r^q_i - r) \cdot \text{diag}(C^{Heur}[i]) \cdot (r^q_i - r)^T}
\]

where \(\text{diag}(C^{Heur}[i])\) is a diagonal matrix with the values from the \(i^{th}\) row of \(C^{Heur}\) as its entries. Figure 11.4 shows an example for a representative \(r\) from a signature of a database object and three query representatives \(r_1, r_2, r_3\) with the corresponding iso-distance lines of the weighted Euclidean distances. In the example, the first query representative has a higher variance in the vertical dimension than in the horizontal dimension. The opposite holds for the second representative while the third one has equal variance in both dimensions. The cost for transforming features from \(r^q_i\) to \(r\) is defined by the local variance information around \(r^q_i\) only.

The resulting ground distance thus adapts the EMD to the user’s feedback by assigning high cost to mass transports in directions where the low variance indicates compact feedback information around a query representative. Similar to the heuristic for histograms, the idea is that compact feedback represents a preference by the user to stay in the area of the feature space.

**Similarity Optimization**

The optimization-based adaptation of the ground distance for feature signatures via \textit{computeGD-Signature-Opt} is built on the same concepts as the optimization
11.2. Relevance Feedback with the Earth Mover’s Distance

Algorithm 11.10: computeGD-Signature-Opt(q, feedback, iter)

1. if \( |\text{feedback}[\text{iter}].\text{rel}| \leq 1 \) then return \( d_{L_2} \);
2. \( C^{\text{Heur}} = \text{computeGD-Signature-Heur-Matrix}(q, \text{feedback}, \text{iter}) \);
3. \( C^{\text{Opt}} = \text{optimize}(C^{\text{Heur}}, q, \text{feedback}) \);
4. return new Distance\( (C^{\text{Opt}}) \);

for the histograms. In fact, the same implementation for the modification of intermediate solutions during the optimization can be used. However, the interpretation is different in one key aspect. As described in the last section, \( C^{\text{Heur}} \) consists of rows with weights for weighted Euclidean distances and not of transformation costs. The updating algorithm modify thus adapts these weights for each individual query representative when it modifies a row of \( C^{\text{Heur}} \). This modification during the optimization process translates to an adaptation of the extents of the iso-distance ellipsoid in the example of Figure 11.4.

11.2.5 Efficient Query Processing

While not the focus of this chapter, it is worthwhile to consider options to speed up both the k-nearest-neighbor computation in Algorithm 11.1 and the ranking computation in the Algorithm 11.2 in order to achieve interactive query processing times. For the k-nearest-neighbor computation within a multimedia database, multi-step retrieval techniques from \([AWS06]\) are well-suited to cope with the changing nature of the transformation cost matrix in the feedback approach. In particular, the lower bound \( LB_{IM} \) which is based on a constraint relaxation of the EMD showed a very good selectivity and good computation times in a prototype implementation. For feature signatures with unequal total weights, \( LB_{IM} \) has to be computed in a slightly adapted manner (i.e., by transforming in the global cost order instead of the dimension-wise order). The indexing technique from Chapter 9 is applicable in the histogram case. Due to their lower-bounding property, these multi-step retrieval techniques return results fast without loss of quality.

A further speed-up can be achieved by trading quality for efficiency. For the optimization process, the maximum iteration count can be decreased resulting
in a less optimized similarity model. Additionally, the EMD computation required to calculate the ranking of the relevant objects within the target function of the optimization can be replaced with an approximate measure such as $LB_{IM}$.

Finally, the CPU-bound ranking computations on the feedback in the optimization algorithm can easily be parallelized for multi-core systems.

As the focus of this chapter is on retrieving satisfactory query results with fewer feedback iterations, only quality-preserving measures were implemented in the evaluation prototype ($LB_{IM}$-based filtering and parallel EMD computation). On a dual Intel XEON E5420 server with 2.5GHz, typical query processing times for the experiments detailed in the evaluation section were around 1.5 seconds.

### 11.3 Experiments

To evaluate whether the adaptation of the EMD ground distance can improve the retrieval effectiveness over the Query-By-Refinement [RT01] method from the literature that relies on query movement alone, a series of automated Relevance Feedback runs were performed. The employed databases and the experimental setup are detailed in Section 11.3.1 with Section 11.3.2 and 11.3.3 giving the results for the adaptation of the EMD on fixed-binning feature histograms and feature signatures.
11.3.1 Setup and Datasets

All experiments were conducted using features extracted from two image databases. The first database (PHOTO) includes 59,896 color photographs and images from a wide variety of themes. Each theme includes 99-100 images. The second database (ALOI) includes 72,000 freely available images of 1,000 objects that were physically rotated around the vertical axis in 5 degree steps [GBS05]. The variation per object is much smaller than the variation per theme in the PHOTO database. In addition, the images in ALOI exhibit a homogeneous black background. For all images, 20-dimensional histograms were extracted using a sampled k-means clustering of features from a 7-dimensional feature space (2 position components, 3 CIE Lab color components, and contrast & coarseness texture components). Signatures with up to 20 components were created using the same clustering method per image instead of per database plus a post-processing step to remove clusters that largely contained noise. For both databases, 20 themes / objects were chosen as relevant for 20 automated feedback runs. The choice was random for the ALOI database. For the PHOTO database, the visual features were often drastically different for images even within single themes. Given the low-level feature extraction process detailed above, themes like “recreation” and “sports” exhibit a very large visual diversity within the themes and significant overlap among the themes. They are thus not suitable for the automated evaluation method used here. To address this issue, the themes were sorted by ascending approximate overlap with other themes and twenty themes with limited overlap were chosen while excluding themes that exhibited hardly any feature diversity. Figure 11.5 show the initial query images that were randomly chosen for the relevant themes / objects. The diversity per relevant PHOTO theme is still vastly greater than the diversity per ALOI object.

Given the query objects, the first feedback iteration was started by retrieving the $k$ closest objects according to the Earth Mover’s Distance with a Euclidean ground distance. Parameter $k$ was set to equal the number of relevant images (99-100 for PHOTO and 72 for ALOI) such that the optimal result could potentially be attained. The $k$ images together with the relevant/non-relevant rating derived from the theme/object information were then used to generate
automated feedback for the next iteration. After each iteration, precision/recall
data was collected by ranking the database up to the point where all relevant
images were found. A precision/recall value of 0.75/0.5 for 72 relevant objects,
for example, means that ranking objects according to their descending similarity
value until \(0.5 \cdot 72 = 36\) relevant objects are found on average produced
\(36/0.75 = 48\) objects total and \(48 \cdot (1 - 0.75) = 12\) non-relevant objects. The
precision/recall curves are computed as described in [MRS08] and averaged
over the 20 queries. The first iteration always starts with a precision value of
1.0 since the 20 initial queries are contained in the database. Given a data-
base (PHOTO or ALOI) and a feature representation (histograms or signatures),
the first iteration is identical for all adaptation approaches (query movement,
heuristic EMD adaptation, and optimization-based EMD adaptation) since no
feedback has been collected up to that point.

The starting temperature \(T^0\) of the optimization process was set to the largest
possible difference in average precision which is
\[
\frac{|\mathcal{I}|}{|\mathcal{R}|} \cdot \sum_{i=|\mathcal{I}|+1}^{|\mathcal{I}|+|\mathcal{R}|+1} \frac{1}{i}
\]
where \(|\mathcal{R}|\) is the number of relevant objects and \(|\mathcal{I}|\) the number of non-relevant objects present
in the feedback. The number of iterations was limited to \(\text{maxIter} = 500\) while
the modification magnitude was set to \(\text{modRowWeight} = 5\%\). Preliminary ex-
periments showed that the optimization on signatures was able to benefit from
a slower temperature decrease compared with the optimization on histograms.
A value of \(\text{coolFactor} = 0.85\) was used for all signatures experiments while 0.8
was used for the histogram experiments.

### 11.3.2 EMD Adaptation for Fixed-Binning Feature Histograms

Figures 11.6 and 11.7 show empirical results for the two histogram databases.
The baseline algorithm (a) with a fixed Euclidean ground distance is contrasted
with the proposed heuristic and optimization-based adaptations in (b) and (c).

For both databases, the adaptation approaches consistently show consider-
able improvements regarding the quality of the query results. The second itera-
tions (i.e., the first EMD adaptation) of both the heuristic and the optimization
outperform even the fifth iteration of the query movement approach by return-
ing fewer irrelevant objects for all recall levels. Furthermore, the optimization
11.3. Experiments

Figure 11.6: Averaged results over 20 runs on the PHOTO histogram database

approach is able to significantly improve on the heuristic approach for iterations three to five.

The diagram series (a) to (c) gives information on full rankings of the databases and shows the overall quality of the adaptation approaches. The precision within the subset of the database that is retrieved in the feedback iterations gives important additional information for assessing the relevance feedback process. For the first $k$ objects, Figure 11.6 (d) shows improvements of more than 50% for the heuristic and more than 100% for the optimization-based adaptation compared with the query movement technique on the PHOTO database. For the homogeneous ALOI database, the baseline algorithm with the fixed ground distance is already able to retrieve a large fraction of all relevant objects. Still,
Figure 11.7: Averaged results over 20 runs on the ALOI histogram database

the similarity adaptation techniques produce 13-21% more relevant objects per iteration as shown in Figure 11.7 (d).

11.3.3 EMD Adaptation for Feature Signatures

Figures 11.8 and 11.9 show results for the adaptation of the EMD on signatures.

The heuristic approach for the EMD on the PHOTO signatures gives consistently better results than the query movement approach (around 7.5% according to Figure 11.8 (d)). As a comparison between Figure 11.6 (c) and Figure 11.8 (c) demonstrates, the optimization-based EMD adaptation is able to successfully exploit the flexibility of both the EMD similarity measure and the signature representation. The baseline algorithm is not able to improve its
results after iteration three whereas the optimization-based EMD adaptation improves with every iteration, leading to $\sim 50\%$ more relevant objects in the result set of the fifth iteration when compared with the query movement approach and nearly three times as many relevant objects compared with its first iteration.

The ALOI signature database shows an unexpectedly poor performance for the baseline algorithm when compared with its histogram counterpart. The overall worse precision values can be explained with the feature clustering used to create the signatures.\(^{†}\) As Figure 11.5 depicts, all images in the database

\(^{†}\)In addition to the overall worse precision values, iteration 4 falls back behind the earlier iterations. For two of the 20 queries, a relevant object is only found at position 3. The individual recall/precision graphs of those two queries thus start with a precision value of $1/3$. For all other queries and iterations on the ALOI database, the first object was always a relevant one.
exhibit large, homogeneous black areas around the borders. The color and texture subclusters in this database result in signature representatives with almost random values for the non-clustered spatial dimension within the border area, which in turn dominate the EMD.

The proposed EMD adaptation approaches overcome this challenge. The statistics-based heuristic leads to low costs for transports in feature space dimensions with high variances. In this way, the adaptation makes the EMD largely ignore the random spatial dimensions for representatives in the border area. This property of the heuristic for signatures leads to substantial improvements regarding the query results as depicted in Figures 11.9 (b) and (d), where even the second iteration shows significantly better precision values than any of the
11.3. Experiments

Figure 11.10: Example for query results after 5 iterations

baseline iterations and relative improvements of up to 80% compared with the baseline for the number of relevant objects returned per iteration.

The optimization-based Relevance Feedback technique still exceeds these effectiveness gains as it takes the objects deemed relevant as well as the objects deemed non-relevant into account when minimizing the discrepancy between the feedback and the similarity measure defined via the adapted EMD. Figure 11.9 (d) shows that the substantial improvements depicted in the precision-recall graph of Figure 11.9 (c) translate to up to twice the number of relevant objects being returned compared with the query movement algorithm that does not alter the EMD.

Lastly, as an example of the images being returned from the PHOTO database, Figure 11.10 shows the first 15 results of the fiftieth relevance feedback iteration for the query image depicted in the bottom left of Figure 11.5. The query consisted of a feature signature for a massive blue door with a light background. The query movement technique (a) gives a mixed result with six non-relevant objects of which the first one is returned at position 2. Among the non-relevant objects are images of a telephone, a penguin, and a power supply. The heuristic approach (b) shows three non-relevant objects with the first one at position 9 only. Given the simple low-level features, the result can be deemed acceptable. The optimization approach is able to return 15 images from the relevant theme of heavy doors in this example. It is noteworthy that the optimization approach had to abstract from the blue color of the door in the query in order to retrieve more relevant objects.
11.4 Summary

This chapter described how the flexibility of the Earth Mover’s Distance can be exploited to explore multimedia databases via a Relevance Feedback process. The key to the adaptation of the EMD is its ground distance, which can be adapted to effectively reflect a query-dependent notion of similarity in the feature space. To that end, an optimization-based extension to the traditional feedback loop and two statistics-based heuristics were proposed. They enable the user to retrieve significantly more relevant objects from the database in fewer iterations compared with existing EMD-based query movement techniques. The optimization-based extension to the feedback loop is general enough to be easily transferred to other adaptable distance functions by defining the parameters that describe the adaptability.
Part V

The Earth Mover’s Distance in Context of Structured Objects
In the literature, the EMD has primarily been applied to assess the similarity between objects that do not possess a strong notion of structure. The last content part of this work presents original research on new ways of applying the EMD in the context of objects such as graphs and time series, where ignoring the order or structure within the objects would be detrimental to the similarity model.

Chapter 12 focuses on the use of the EMD to assess the similarity between medium-sized graphs based on their structure. The approach considers the connectivity degree of nodes on paths through a graph and maps them to signatures to be compared using the EMD.

Chapter 13 presents original research (cf. [AWK^09]) that is motivated by the desire to use the EMD as a ground distance for the DTW on time series of multimedia objects. As a preparation for the use of $DTW_{EMD}$ for similarity search in multimedia databases, the high number of ground distance computations for each DTW computation is reduced by exploiting information from the filter step of DTW-based retrieval algorithms.
Chapter 12

Feature-Based Graph Similarity
with the Earth Mover’s Distance

Graph structures are utilized to represent a wide range of objects including naturally graph-like objects such as molecules and derived graph structures such as connectivity graphs for region-based image retrieval. This chapter proposes to extend the applicability of the EMD to graph objects by deriving a similarity model with a representation of structural graph features that is compatible with the feature signatures of the EMD. The aim is to support the search for a graph in a database from which the query graph may have originated through limited structural modification. Such query graphs with missing or additional vertices or edges may be the result of natural processes of decay or mutation or may stem from measuring methods that are inherently error-prone, to name a few examples.

12.1 Introduction

In the absence of a canonical representation of graphs, deciding if two graphs are isomorph (i.e., identical but for a renaming of the vertices) is a computationally expensive task. Its generalization, the subgraph isomorphism problem, is known to be NP-complete. When attempting to find all graphs in a database that contain a subgraph that is isomorph to some query graph, it is possible to
use techniques similar to the lower-bounding filtering reviewed in Section 3.1 to quickly rule out some candidates and refine the rest with the computationally expensive exact matching. For example, the GraphGrep approach indexes labels along paths within a graph to perform the filtering [SWG02].

For similarity search, deciding whether (sub)graphs are isomorph does not suffice. In the case of the two graphs not being identical, similarity search requires an assessment of the degree to which the graphs in question differ from another.

The comparison of two graphs can be performed by directly considering the structure of the graphs. This approach is, for example, taken by the graph edit distance [BA83] that calculates how many transformations have to be performed to turn one graph into the other, and also by measures that consider common subgraphs or the size of the largest common subgraph [BS98]. While these measures are suitable for small graphs and for graphs with limitations regarding their structure and/or the operations that may be performed (e.g., the degree-2 edit distance for connected, undirected, acyclic trees [ZWS96]), even medium-sized general graphs quickly lead to a query processing time that is bound to overburden the patience of the user.

Akin to content-based image retrieval, feature-based graph similarity models instead derive (approximate) structural information from the graphs and assess the similarity of the graphs based on these features. For example, so-called spectral approaches [Ume88, LWH03] compare graphs based on an eigen-decomposition of the adjacency matrix. The model presented in [PM99] compares two graphs by computing the difference in the number of nodes that have a given connectivity degree. The latter is the basis for the generalized approach described in this chapter. Connectivity information collected along paths in graphs is collected and represented in a way that is compatible with feature signatures (cf. Definition 1.2), thus allowing graphs to be flexibly compared using the EMD. In a recent related approach, graphs derived from images have been compared using the EMD [GXTL08]. However, the approach uses the EMD to compare the direction of edges/lines that occur in the graph and thus requires the vertices to have a spatial position. The approach described here does not make such an assumption.
12.2 Preliminaries

The basic graph-related definitions for concepts used in the rest of the chapter are given in this section with a small example on page 204.

A general graph with at most one edge from one vertex to another is defined via its set of vertices and its edge relation.

**Definition 12.1 (Graph)**

A graph $G$ of size $m$ is a tuple $G = (V, E)$ with vertices $V = \{v_1, \ldots, v_m\}$ and edges $E \subseteq V \times V$.

If a graph does not have single-vertex loops (i.e., the edge relation is irreflexive) and is undirected (i.e., the edge relation is symmetric), it is called simple.

**Definition 12.2 (Simple Graph)**

Given a graph $G = (V, E)$, $G$ is simple iff for all $v, w \in V$

$$(v, w) \in E \iff (w, v) \in E \text{ and } (v, w) \in E \Rightarrow v \neq w.$$  

All graphs examined in the remainder of this chapter are assumed to be simple.

If all vertices of a graph are connected to all other vertices of the graph by a series of edges, it is called connected.

**Definition 12.3 (Connected Graph)**

Given a graph $G = (V, E)$, $G$ is connected iff for all $v, w \in V$:

$$(v \neq w) \Rightarrow (v, w) \in E$$

$$\forall \left( \exists v_i, \ldots, v_{i+1}', w \in V : \ (v, v_i) \in E \land (v_i, w) \in E \land 1 \leq j \leq m' - 1 : (v_j, v_{j+1}) \in E \right).$$

The graphs in the database are assumed to be connected in this chapter. A query graph with missing vertices or edges may however break down into several non-connected components.

The degree of a vertex in a graph is the number of other vertices it is directly connected to.
**Definition 12.4 (Vertex Degree Function)**

Given a graph $G = (V, E)$, the outgoing vertex degree function $\delta^G : V \rightarrow \mathbb{N}_0$ for $G$ is defined by

$$\delta^G(v) = |\{w \in V |(v, w) \in E\}|.$$  

The ingoing vertex degree function can be defined analogously. For the simple graphs of this chapter, the two functions are identical and thus do not have to be differentiated here.

The example graph $G_1$ in Figure 12.1(a) is a simple, connected graph with 5 vertices and 6 edges. Figure 12.1(b) gives the adjacency matrix of $G_1$ where an entry of 1 indicates the existence of an edge while an entry of 0 indicates the absence of an edge between two vertices. As a result of Definition 12.2, the diagonal entries are all zero and the matrix is symmetric. The degree of a vertex equals the row sum in the adjacency matrix. Vertices $v_1$, $v_3$, and $v_4$ have degree 2 while vertices $v_2$ and $v_5$ have degree 3. The graph $G_2$ is neither simple (due to the loop at $v_5$) nor connected (due to having two separate components).

### 12.3 Graph Similarity Model

In order to find graphs in a database that might be related to a query graph through a process of decay, mutation, or generally structural change, a representation of statistical graph features is proposed in Section 12.3.1 and distance measures suitable for the feature representation are given in Section 12.3.2. The similarity of two graphs can be assessed by combining these two parts.
12.3.1 Graph Feature: Degree Co-Occurrence Multisets

A representation of graph features that encodes structural information is required for detecting small structural changes between graphs in a feature-based approach. In this section, statistical features of the vertices that occur in the graphs and their connectivity relationship are discussed. In the simplest form, a graph can be represented by the distribution of the degrees of its vertices as in [PM99]. However, by looking at each vertex separately, one of the core concepts of graphs is ignored. Graphs are useful as they model relationship information between the vertices. Thus, this section proposes to utilize statistical information on the co-occurrence of vertices. In this way, the feature representation encodes which kinds of vertices are connected within a graph – and how frequent this coupling occurs. The co-occurrence concept can be generalized by looking at occurrences along paths in the graph and noting which kinds of vertices occur close to each other / in sequence. In the following definitions, the generalized co-occurrence concept is formally introduced on the basis of vertex degrees as this information is common to all graphs. If other categorical information (e.g., vertex class labels) is available, the approach could be adapted to incorporate that information.

**Definition 12.5 (Simple Vertex Path)**

With $G = (V, E)$ as a graph, the $(m + 1)$-tuple $(v_{i_0}, \ldots, v_{i_m}) \in V^{m+1}$ is a simple (non-looping) vertex path of length $m$ in $G$ iff

$$\forall 0 \leq j < j' \leq m : v_{i_j} \neq v_{i_{j'}}$$

and

$$\forall 0 \leq j < m : (v_{i_j}, v_{i_{j+1}}) \in E.$$  

The set of all simple paths of length $m$ in $G$ is denoted as $P^G_m$.

For the cases of path lengths $m = 0$ and $m = 1$, sets $P^G_0$ and $P^G_1$ equal the set of vertices $V$ and the set of edges $E$. Using the set of simple paths of length $m$, a co-occurrence multiset of degree $m$ captures the frequencies of vertex class (here, vertex degree) sequences.
Definition 12.6 (Vertex Degree Co-occurrence Multisets)
With $G = (V, E)$ as a graph, the Vertex Degree Co-Occurrence Multiset $D^G_m$ of degree $m$ for graph $G$ is defined as a tuple

$$D^G_m = (D^G_m, f^G_m)$$

where

$$D^G_m = \{(\delta^G(v_{i_0}), \ldots, \delta^G(v_{i_m})) \mid (v_{i_0}, \ldots, v_{i_m}) \in P_m \}$$

is the set of all vertex degree sequences occurring on paths of length $m$ in $G$ and

$$f^G_m(dg_0, \ldots, dg_m) = \left| \{(v_{i_0}, \ldots, v_{i_m}) \in P_m \mid \forall 0 \leq j \leq m : dg_j = \delta^G(v_{i_j}) \} \right|$$

the frequency function of such sequences in $G$.

The set $D^G_m$ abstracts from individual vertices by only considering their type (i.e., vertex degree in this example). The degree $m$ of the multiset is not related to the degree of the vertices in the graphs but only to the length of the examined paths.

As an example, the graph $G_1$ in Figure 12.1(a) has five paths of length $m = 0$ (i.e., $P^G_0 = V = \{v_1, \ldots, v_5\}$). The occurring vertex degrees are $D^G_0 = \{(2), (3)\}$ with frequencies $f^G_0(2) = 3$ and $f^G_0(3) = 2$. For $m = 1$, there are six paths (i.e., one per edge). The combinations of vertex degrees occurring along those paths are $D^G_1 = \{(2, 2), (2, 3), (3, 2), (3, 3)\}$. The frequencies of those paths are $f^G_1(2, 2) = 2, f^G_1(2, 3) = 4, f^G_1(3, 2) = 4, \text{ and } f^G_1(3, 3) = 2$. Generally, the number of paths grows for greater lengths $m$ and for denser graphs. The set of vertex degree sequences $D^G_2$ is of cardinality 6 and $D^G_3$ of cardinality 8. The experiments in Section 12.4 show good results for $m$ as low as 2. For greater lengths, techniques such as random path sampling could be applied to speed up the feature extraction process.

Figure 12.2 shows a visualization of the co-occurrence multisets $D^G_1$ (on the far left) and $D^G_2$ (on the far right) in the form of bubble charts. The $x$, $y$, and $z$ axes denote the degree of the first, second, and third vertex on a path in $G$. The size of the bubbles is proportional to the frequency of the according vertex degree sequences that is also denoted inside the bubble. For $m = 1$, the short
12.3. Graph Similarity Model

![Graph Representation](image)

Figure 12.2: Visualization for the multiset feature representation of a graph G

arrows next to the graph in the middle of the figure show all paths (i.e., edges) that contribute to the multiset $D_G^1$. The long arrow in the upper section of the graph shows a path that contributes to the bubble at coordinate 3-2-3 in the far right of the figure representing $D_G^2$.

With the above definitions, a co-occurrence multiset can be associated with each graph in the database and with the query graph. The similarity of the graphs can then be assessed in terms of the co-occurrence multisets that contain statistical information on the structure of the graphs.

### 12.3.2 Similarity Measure

**Element-Wise Multiset Comparison**

As the multisets have a finite domain $DS$ for any finite graph, element-wise comparisons can be applied. By treating the multisets as a sparse representation of high-dimensional vectors, norm-based distance measures such as the $L_p$ distances can be adapted to compare two graphs.

**Definition 12.7 ($L_p$ Distance on Vertex Degree Co-Occurrence Multisets)**

Given two graphs $G_1$ and $G_2$ with associated vertex degree co-occurrence multisets $D_{m}^{G_1} = (DS_{m}^{G_1}, f_{m}^{G_1})$ and $D_{m}^{G_2} = (DS_{m}^{G_2}, f_{m}^{G_2})$ according to Definition 12.6, the $L_p$
The Earth Mover’s Distance (EMD) is a measure of the dissimilarity between two distributions. In the context of graph similarity, the EMD is used to compare the co-occurrence multisets of graphs. The distance between the two multisets is defined as

\[
d_{L_p}(D^{G_1}_m, D^{G_2}_m) = \left( \sum_{ds \in (DS^{G_1}_m \cap DS^{G_2}_m)} |f^{G_1}_m(ds) - f^{G_2}_m(ds)|^p \right) + \sum_{ds \in (DS^{G_1}_m - DS^{G_2}_m)} |f^{G_1}_m(ds)|^p + \sum_{ds \in (DS^{G_2}_m - DS^{G_1}_m)} |f^{G_2}_m(ds)|^p \right)^{1/p}.
\]

In the case of \( m = 0 \) and \( p = 1 \), the similarity model reflects the one of [PM99] where graphs are compared using simple vertex degree histograms and the Manhattan distance.

**Transformation-Based Multiset Comparison**

The co-occurrence multisets are a close match to the signatures that the EMD takes as its input. The similarity of two graphs can be assessed using the EMD by defining a transformation from the multisets to the signatures of the EMD.

**Definition 12.8 (Feature Signatures of Graphs)**

Given a graph \( G \) with an associated vertex degree co-occurrence multiset \( D^G_m = (DS^G_m, f^G_m) \), the feature signature \( s^G_m \) of \( G \) for comparison with the Earth Mover’s Distance is defined as

\[
s^G_m = \left\{ (r, w) \mid r \in DS^G_m \land w = \frac{f^G_m(r)}{|DS^G_m|} \right\}.
\]

The cost for transforming one degree sequence into another one can be defined via a ground distance function. In the simplest case, the sequences can be treated as vectors from \( \mathbb{N}_0^m \) and compared using \( d_{L_p} \) from Definition 1.6 on page 16. In this way, a degree sequence that deviates from another for example by starting with a degree of 3 instead of 4 will induce a lower transformation cost than one that starts with 1 instead of 4. Distance measures such as the Edit Distance, which take the sequential character of the representatives \( r \) into
account, could also be employed. For undirected graphs, the fact that each sequence of vertex degrees appears twice in both directions should be accounted for by adjusting either the signature definition or the ground distance.

12.4 Experimental Evaluation

12.4.1 Synthetic Graph Generation

For the experiments shown here, a number of synthetic graph databases of differing cardinalities were created. A synthetic graph of size $m$ can be generated through a randomized process in a number of ways. The simplest is to assume the edge relation to be independent and identically-distributed (IID). The existence of an edge between any two of the $m$ vertices has a fixed probability $p$. This graph generating process is called the Erdős-Rényi model [ER59] and results in a binomial vertex-degree distribution where a large number of vertices has a degree of medium value with smaller and greater degrees being less likely. As discussed in [AB02], graphs found in real applications often do not follow a binomial vertex-degree distribution resulting from the IID edge relation. An increasing number of graphs have instead been found to follow a power-law distribution where a large majority of vertices have a small degree with very few vertices having a large degree (i.e., they function as hubs). Examples listed include small graphs such as predator-prey relationship of animals in closed habitats with a low 3-digit number of vertices (i.e., species) and large graphs such as the world wide web linking graph with millions of vertices (i.e., web pages).

The synthetic graph databases used in the experiments were generated using the method detailed in [VL05] based on sequences of vertex degrees following a power-law distribution with modifications to ensure that the graphs are connected and simple. Figure 12.3 illustrates the resulting vertex degree distribution for ten random graphs as used in the experiments (size 100, vertex degrees between 1 and 15, and an average degree of 3) and contrasts them to according graphs following the simple Erdős-Rényi model.
12.4.2 Results

In the first set of experiments, 100 graphs were randomly chosen from the database as the basis for 15 query graphs each that represent different levels of structural deviation regarding the edge relation. For each of the 15 levels, a random edge was either inserted or deleted with equal probability. Not accounting for edges that may have been deleted and consecutively been added again, up to 10% of the edge relation may have been changed in this process.

The vertical axis of Figure 12.4 shows how often the graph on which the query graph was based was identified as the most similar one out of all 1000 graphs in the database. A greater path length (denoted as PL in the figure) for the vertex degree co-occurrence multisets results in a similarity model that is more robust with regard to the structural change for this experiment. The greater the structural difference, the more can the multisets based on longer paths distinguish themselves from those of lower degree. The Manhattan distance on simple vertex degree histograms (cf. [PM99]) is always outperformed by the multisets of higher degree (i.e., based on longer paths) in this experiment. The EMD with a Manhattan ground distance slightly outperforms the Manhattan distance for equal path lengths. The EMD for path length zero is not plotted here, as the results equal those of the Manhattan distance in the case of
12.4. Experimental Evaluation

Figure 12.4: Adding/deleting edges at random for $|DB| = 1000$

Figure 12.5 shows that the higher degree multisets are also less influenced by the cardinality of the database. Even though the database size on the right is four times the size of the database on the left, the number of times that the original graph from the database is not identified as the most similar one to the query graph only slightly increases from 9 out of 100 to 14 out of 100 for the EMD with path length two. The degree histogram approach jumps from 10 out of 100 to 28 out of 100 for the same increase in database size.

The two figures 12.6 and 12.7 show the results of according experiments when considering structural change that is not limited to the edge relation. Instead, random vertices were removed together with their adjacent edges. As is to be expected due to the greater level of structural change, all approaches show a faster decrease of the precision with which they can identify the original graph in the database. Greater path lengths still produced better results in these experiments while the EMD with a Manhattan ground distance was only able to outperform the normal Manhattan distance for more severe levels of structural change in this case.
Figure 12.5: Adding/deleting 10 edges at random

Figure 12.6: Deleting vertices at random for $|DB| = 1000$
12.5 Summary and Outlook

This chapter showed how complex data objects in the form of graphs can be compared using the EMD by defining a suitable representation of graph features that capture statistical information regarding the structure of the graphs. In this way, it is possible to identify graphs that resulted from some other graph through a process of structural change without having to resort to typically very expensive similarity measures that directly take the graph structure into account.

The general viability of the approach was shown using a Manhattan ground distance for the EMD together with vertex degrees as the sole information regarding the vertices. For this ground distance, the projection-based lower bound of Section 7.5.1 and the EMD-L1 algorithm from [LO07] could be employed to speed up retrieval. While the results using this simple ground distance were also generally good, the Manhattan ground distance potentially limits the benefits of longer co-occurrence sequences that are used as signature component representatives for the EMD. Other ground distances that take the sequence character of the feature representatives (i.e., sequences of vertex degrees in this case) into account may present an opportunity to further improve the results.

Figure 12.7: Deleting 5 vertices at random
Chapter 13

Reducing the Number of DTW Ground Distance Computations

The desire to use a transformation-based distance measure as a ground distance for another transformation-based distance measure is the motivation for the development of techniques that reduce the number of ground distance computations in a similarity search system since the determination of each ground distance is computationally expensive in this scenario. In this chapter, a concept called *Anticipatory Pruning* is proposed for speeding up DTW-based retrieval on both univariate and multivariate time series in this manner. Intuitively, Anticipatory Pruning reduces the number of ground distance computations by anticipating the DTW refinement result through exploiting information from previous filters in a multi-step filter-and-refine algorithm.

13.1 Introduction

Time series or feature sequences (cf. Definition 1.3) record the value of one or more attributes as they change over a span of time. For example, temperature measurements from several spatial locations of observation could be recorded at each full hour for each full year enabling year-to-year climate change analysis. Another example where the value of a number of attributes is recorded are digital videos. In this case, each point in time corresponds to a frame of the
video and the values in the time series are either the raw pixel values of each frame or a representation of the features that describe characteristic properties of the frame (e.g., color histograms). When comparing two time series with complex feature representations such as video sequences, the effectiveness of the overall model depends on both the similarity measure used to compare the feature representation for two points in time and on the similarity measure used to aggregate the point-in-time similarity measures to an overall similarity score for the time series. As discussed in the previous chapters, the Earth Mover’s Distance is well suited for comparing histogram-based feature representations. For time series retrieval, the DTW distance is a prime candidate when effectiveness is of concern. Integrating the two measures to form $DTW_{EMD}$ as proposed by the video retrieval model in [AK09] promises to yield a high-quality similarity measure for time series of histogram-based features representations. However, the potential quality comes at a high computational cost. For each entry in the valid band of the cumulative distance matrix (cf. Section 6.3), an Earth Mover’s Distance has to be computed. Thus, a reduction of the number of distance matrix entry computations during query processing leads to a significant performance improvement for similarity search on time series using DTW with a computationally expensive ground distance.

The Anticipatory DTW (aDTW) algorithm proposed here for the purpose of reducing the number of ground distance computations for DTW is lossless for a certain class of lower-bounding filters that are characterized in Definition 13.3. Widely used state-of-the-art approaches belong to this class. Additionally, as aDTW efficiently reuses information from previous filter steps, it incurs very little computational overhead. The experiments in Section 13.3 demonstrate the performance improvements that can be achieved. As aDTW is orthogonal to existing lower-bounding filtering, indexing, and dimensionality reduction techniques, it can be flexibly combined with such techniques for additional speed-up.

### 13.1.1 Related Work

Techniques to speed up the retrieval of time series based on DTW can be categorized as either correct or approximate with respect to the result set they deliver.
Approximate techniques that do not return the exact same result set that a simple scan with DTW would produce include [YJF98, CFY03, CKHP02, SC04, SC07, APP+08]. In [YJF98] an approximate embedding of DTW into Euclidean space is proposed. This technique is extended to a technique based on the Haar wavelet transform for fewer false positives [CFY03], but possibly more false negatives. Iterative Deepening DTW [CKHP02] computes different levels of dimensionality reduction from piecewise linear approximations. Using a probabilistic model based on sample approximation errors, time series are either pruned or compared at a finer level of approximation. The FastDTW approach [SC04, SC07] computes approximations of the warping path at different levels of granularity. The recent embedding-based subsequence matching technique [APP+08] hashes subsequences of the original time series to vectors based on their DTW distance to reference time series. Subsequences that are identified as potentially similar to the query are then refined using the DTW. All of these techniques provide efficiency gains by sacrificing the correctness of the result as they are approximate in nature.

Correctness of the result is guaranteed by lower-bounding filter-and-refine techniques. The time series representation of the Piecewise Aggregate Approximation (PAA) method is composed of averages of consecutive values along the time axis [KCPM01]. The Adaptive Piecewise Constant Approximation adapts the number of values that are averaged to reduce the approximation error [KCPM00]. For these representations, lower-bounding distance functions for DTW have been proposed. Another lossless approach is based on four characteristic values of time series: their starting and ending value, their minimum, and their maximum [KPC01]. \(LB_{Keogh}\) provides a lower-bound that uses the warping path constraint to compute an envelope on the warping cost for PAA segments [Keo02]. This approach has been extended to 2-dimensional envelopes around trajectories in [VHGK03, RM02]. Improved versions of the envelope technique have been proposed in [ZS03] and [ZW07]. The Fast search method for Time Warping (FTW) uses different resolutions to gradually refine the optimal warping path [SYF05]. Using Early Stopping, only the warping paths with a DTW distance not exceeding some pruning distance are evaluated. This technique is also known as Early Abandoning [KWX+06].
The technique proposed here can be integrated with these lower-bounding distance functions to improve the overall retrieval performance. More details on the integration are given in Section 13.2.8.

13.2 Anticipatory DTW

The multi-step approach proposed in this chapter extends the classic filter-and-refine algorithms for DTW and is called Anticipatory DTW or aDTW for short. It is based on a number of properties of existing lower-bounding filters (Sections 13.2.4 to 13.2.6) and makes use of an Anticipatory Pruning Distance APD (Section 13.2.7). Theorem 13.3 shows that APD is a lower bound for the DTW distance, which implies that aDTW is complete with regard to its result set.

13.2.1 Structure of aDTW

The general structure of aDTW is similar to the multi-step filter-and-refine frameworks used in chapters 8 and 9. However, there are three design characteristics specific to aDTW (illustrated by Figure 13.1).

1. There are at least two filters in the chain – the first one is taken from the literature and the second one is the proposed Anticipatory Pruning.

2. The computation of APD is performed in a number of incremental steps.
After each step, a pruning decision is made.

3. APD is not independent of the filter from the literature but efficiently exploits its information.

Traditional filter-and-refine algorithms compute the full DTW if the filter does not prune the candidate. The pruning decision is made by comparing a lower-bounding filter distance with a pruning threshold given, for example, by the range of a range query or by distances encountered so far during k-nearest-neighbor search. An existing improvement of this technique is known as Early Stopping [SYF05] or Abandoning [KWX06], where the computation of DTW in the refinement step is stopped as soon as an intermediate result exceeds the pruning threshold. The aDTW concept proposed here goes beyond Early Stopping.

If the first filter cannot prune a candidate, APD₁ is computed. If APD₁ does not allow for pruning either, APD₂ is computed. This process is continued until either pruning is possible or APDₘ has been computed. In a step 1 ≤ i ≤ m, APDᵢ is based on the DTW computation up to column i of the cumulative distance matrix from Chapter 6 – just as in Early Stopping. Additionally, APDᵢ anticipates the steps i + 1 to m by reusing information from the first filter as indicated by the arrows at the top of the figure. Thus, APD has an improved approximation quality compared with Early Stopping alone. The computation of APDₘ includes all information required to derive the DTW distance. No further refinement step is necessary.

13.2.2 Preliminaries

As some of the theorems and proofs in this chapter depend on the warping path within a given valid band for DTW (cf. Definition 6.2), the valid band and the warping path shall be defined here along with some elementary properties of warping paths. The Sakoe-Chiba band is assumed for ease of discussion, but the technique can be easily adapted to other bands as well.

**Definition 13.1 (Valid matrix band for k-band Dynamic Time Warping)**

The valid cells with respect to a band parameter k in the cumulative DTW distance
matrix of size $m_1 \times m_2$ (w.l.o.g. $m_1 \leq m_2$) for two time series $s$ and $t$ of length $m_1$ and $m_2$ are

$$\text{band} = \bigcup_{1 \leq j \leq m_2} \text{band}_j$$

with $\text{band}_j$ as the valid cells in column $j$:

$$\text{band}_j = \{(i, j) \mid 1 \leq i \leq m_1, \ |i - \left\lfloor \frac{j \cdot m_1}{m_2} \right\rfloor | \leq k\}$$

**Definition 13.2 (Warping path)**
The warping path $P$ for two time series $s$ and $t$ of length $m_1$ and $m_2$ is defined as a set of matrix cells

$$P = \{p_1, \ldots, p_l\} \text{ with } p_1 = (0,0), \ p_l = (m_1, m_2).$$

**Proposition 13.1 (Properties of warping paths)**
For any two $p_i = (a, b), p_{i+1} = (c, d)$ of a warping path $P$ in a cumulative distance matrix $D$ of $\text{DTW}_{g,d}$, the following holds:

1. monotony: $c - a \geq 0 \land d - b \geq 0$
2. continuity: $c - a \leq 1 \land d - b \leq 1$
3. alignment: $D[c, d] = gd(s_c, t_d) + D[a, b]$

A warping path is monotonic in the sense that the time does not go backwards. The path is continuous in that there are no omitted points in time; successive path elements correspond to consecutive points in time. The alignment property ensures that each cell $p_i$ arose from its direct predecessor $p_{i-1}$.

**13.2.3 The Idea of aDTW**
The Anticipatory Pruning Distance APD is based on three properties.

1. DTW computation is incremental, i.e., the entries in the cumulative distance matrix increase in value with the sequence length.
2. Many existing lower-bounding filters for the DTW distance can be characterized as piecewise, i.e., a subsequence of the filter computation is a valid filter for a subsequence of a time series.

3. DTW is reversible, i.e., computing the warping path from the beginning to the end is the same as doing so from the end to the beginning.

The Anticipatory DTW algorithm uses these three properties by calculating piecewise filter distances on reversed time series in the filter step of the framework depicted in Figure 13.1. During the incremental computation of APD (non-reversed), the information from the piecewise filter serves as a lower-bounding anticipatory estimate of the DTW parts yet to be calculated. The exploitation of this additional information allows for stopping the calculation sooner than would have been possible without the anticipatory part. If no stopping is possible, the last step of computing APD corresponds to the last step of computing the DTW distance.

The proposed aDTW is highly flexible in that it can be combined with many existing lower-bounding filters. It is easy to integrate with these filtering approaches, as all the information required for Anticipatory Pruning can readily be derived from existing filter calculations. Moreover, the pruning capability of aDTW comes at hardly any additional cost, as the filter distances have already been computed in the filter step.

In the following sections, the properties important for aDTW are discussed.

13.2.4 Incremental Computation of DTW

Anticipatory Pruning uses filter information for additional pruning checks. These checks are based on the proposed anticipatory estimate and the fact that DTW is incremental.

**Theorem 13.1 (DTW is incremental)**

For any cumulative distance matrix $D \in \mathbb{R}^{m_1 \times m_2}$ as defined in Definition 6.3, the column minima are monotonically non-decreasing:

\[ \forall 1 \leq (y - 1) < y \leq m_2 : \min_{1 \leq i \leq m_1} \{D[i, y - 1]\} \leq \min_{1 \leq i \leq m_1} \{D[i, y]\}. \]
Reducing the Number of DTW Ground Distance Computations

Proof 13.1
The theorem follows from the definition of the cumulative distance matrix: \( D[i, j] = gd(s_i, t_j) + \min\{D[i - 1, j - 1], D[i, j - 1], D[i - 1, j]\} \). Let \( i^* \) be the smallest row index such that \( D[i^*, y] \) is minimal in column \( y \). Due to \( gd(s_{i^*}, t_y) \geq 0 \) and \( D[i^* - 1, y] > D[i^*, y] \), the value of \( D[i^*, y] \) cannot be based on \( D[i^* - 1, y] \) in the same column. Thus, it must either be based on \( D[i^* - 1, y - 1] \) or on \( D[i^*, y - 1] \). W.l.o.g., let it be based on \( D[i^*, y - 1] \). Then \( gd(s_{i^*}, t_y) \geq 0 \) implies that \( D[i^*, y - 1] \leq D[i^*, y] = \min_{1 \leq i \leq m1}\{D[i, y]\} \). The minimum in column \( y - 1 \) can thus be no larger than the minimum in column \( y \).

The value \( D[m1, m2] \) that represents the DTW distance is trivially lower-bounded by the minimum of column \( m2 \). Due to Theorem 13.1, it is also lower-bounded by the non-decreasing sequence of column minima. This property allows for Early Stopping/Abandoning of the DTW computation. As soon as one of the column minima exceeds the pruning threshold, the complete DTW does not have to be computed as the final DTW will exceed the threshold, too.

The definition of APD will go beyond column minima as it adds anticipatory pieces of filter information for a closer estimate of the DTW. Entries \( D[i, j] \) in the cumulative distance matrix correspond to the minimum cost of possible warping paths between subsequences \( s[1 : i] \) and \( t[1 : j] \). The aim is to find a lower bound for the cost of the remaining warping path that has not yet been computed. Provided that a lower bound for the whole warping path can be decomposed such that one piece serves as a lower bound on the remaining postfix of the warping path, this piece can be combined with the column minimum for an overall estimate that will be defined as the Anticipatory Pruning Distance.

13.2.5 Piecewise Filter Computation for DTW
Anticipatory Pruning can also be understood as an incremental refinement of a filter with DTW distance information. As illustrated in Figure 13.2, the cumulative distance matrix for DTW computation is filled as usual. For example, columns 1 to 5 have been computed in the leftmost figure. The column minimum in the fifth column is a lower bound for the DTW distance. Up to this point, only partial time series have been accounted for (namely up to the sev-
enth entry for the first and up to the fifth entry for the second time series). For
the remaining subsequences of the time series, starting in the sixth column, APD
includes an estimate derived from the preceding filter step (cf. Figure 13.1). The
estimate is a piece of filter information corresponding to subsequences of time
series starting in the last seven columns.

This estimate on partial time series alignments requires that the filter be
*piecewise*, i.e., decomposable into a series of lower bounds for all time series
prefixes of increasing length.

**Definition 13.3 (Piecewise DTW lower bound)**
A piecewise lower-bounding filter for the DTW distance between two time series $s$
and $t$ of length $m_1$ and $m_2$ is a set $\{p_{f_0}, ..., p_m\}$ with the following property:

$$p_{f_0}(s, t) = 0$$
$$\forall 0 < j \leq m_2 : p_{f_j}(s, t) \leq \min_{(i, j) \in \text{band}_j} DTW(s[1 : i], t[1 : j])$$

Intuitively, a piecewise lower-bounding filter can be decomposed into a se-
ries of lower bounds for all possible partial DTW warping paths that start at the
beginning of both time series and end in a respective column. The piecewise
property is not a major constraint, as many existing lower-bounding filters for
DTW can be decomposed in such a manner as will be shown in Section 13.2.8.
13.2.6 Reversible Computation of DTW

The piecewise property of a DTW lower bound and the incremental computation property of DTW together do not suffice to combine pieces of the filter with partial DTW computations. To derive an overall lower bound, partial DTW computations up to some column \( j \) of the cumulative distance matrix have to be combined with piecewise filter information for the remaining columns \((j+1), \ldots, m_2\) corresponding to postfixes of the time series. However, the piecewise filter only provides information for prefixes of the time series. Fortunately, DTW computation is reversible, i.e., computing the distance for the reversed time series (from the end to the beginning) yields exactly the same result as computing it for the original time series.

**Theorem 13.2 (DTW is reversible)**

For any two time series \( s = \langle s_1, \ldots, s_{m_1} \rangle \) and \( t = \langle t_1, \ldots, t_{m_2} \rangle \) and for according reversed time series \( s^- \) and \( t^- \) it holds:

\[
DTW(s, t) = DTW(s^-, t^-)
\]

**Proof 13.2**

Assuming that this property does not hold and to the contrary (w.l.o.g.):

\[
DTW(s, t) < DTW(s^-, t^-)
\]

Let \( \{p_1, \ldots, p_{l_1}\} \) and \( \{p_{l_1}^-, \ldots, p_{l_2}^-\} \) be warping paths with minimal cost for the non-reversed and for the reversed time series. Since reversing path \( \{p_1, \ldots, p_{l_1}\} \) yields a valid warping path for the reversed time series (properties monotony, continuity, and alignment of Definition 13.2 follow from the according properties of the non-reversed path) and since its cost remains unchanged and equals \( DTW(s, t) \), it follows that \( \{p_{l_1}^-, \ldots, p_{l_2}^-\} \) with cost \( DTW(s^-, t^-) \) cannot be a warping path with minimal cost. As this contradicts the assumption, DTW is reversible.

Reversibility is important for the proposed Anticipatory Pruning, as it allows for deriving a lossless estimate of the DTW distance from any piecewise lower-bounding filter. By reversing the order of time series during the first filter com-
putation, the same filter distance is obtained as if the filter was computed on the original time series. However, the pieces of the filter computation from the end of the time series backwards to the current point of DTW computation may now be used as an estimate of the remaining path cost. As both this estimate and the column minima lower-bound the DTW on the according subsequences, APD as the sum of those two components is lower-bounding DTW on the complete time series. This statement will be formally proven in Theorem 13.3 after the definition and an example for the Anticipatory Pruning Distance APD are given.

13.2.7 Anticipatory Pruning Distance

Anticipatory DTW includes a sequence of pruning checks. After every column of the cumulative distance matrix that has been computed, piecewise filter information is added to yield an overall lower bound for the exact DTW distance. It can therefore be seen as a series of \( m_2 \) filters for time series \( s \) and \( t \) of length \( m_1 \) and \( m_2 \), where the final step provides the information for the actual DTW distance. Formally, the proposed Anticipatory Pruning Distance is defined as follows:

Definition 13.4 (Anticipatory Pruning Distance (APD))

Given two time series \( s \) and \( t \) of length \( m_1 \) and \( m_2 \), a cumulative distance matrix \( D \) as defined in Definition 6.3, a piecewise lower-bounding filter \( pf \) for the reversed time series \( s^{-} \) and \( t^{-} \) as defined in Definition 13.3, and a value \( j \in \{1, \ldots, m_2\} \), the \( j \)th step of the Anticipatory Pruning Distance APD for the k-band DTW distance between \( s \) and \( t \) is defined as

\[
APD_j(s, t) = \min_{1 \leq i \leq m_1} \{D[i, j]\} + pf_{m_2-j}(s^{-}, t^{-}).
\]

The Anticipatory Pruning distances are thus a sequence of DTW column minima with added pieces of filter information for the remainder of the time series. During computation step \( j \), the column minimum of the partial path between \( s[1 : i] \) and \( t[1 : j] \) according to DTW is combined with the anticipatory information for the rest of the alignment available from the lower-bounding filter. The resulting estimate of the entire cost of the warping path in step \( j \) is \( APD_j \).
Figure 13.3 gives an example for the Anticipatory Pruning Distance. The left part of the figure shows two univariate time series \( s \) and \( t \) together with their cumulative distance matrix. For reasons of simplicity, this example uses an \( L_1 \)-based variant of the \( LB_{Keogh} \) lower bound for the filter \( pf \). The relationship between the DTW distance (35 as seen in the top right corner of the matrix), the pruning threshold (chosen as 22), and the components of \( APD_j \) are depicted in the right part of the figure.

The filter distance \( pf \) between the reversed \( s^- \) and \( t^- \) yields the following piecewise results: 0,7,...,18. Thus, the filter step distance is 18. To make the computation of the Anticipatory Pruning Distance more intuitive, the order of the piecewise filter results are reversed in the figure. As the filter distance of 18 does not exceed the pruning threshold of 22, \( t \) cannot be pruned by the filter and DTW would normally have to be computed now (potentially with Early Stopping). Instead, aDTW starts with filling the cumulative distance matrix for DTW(\( s, t \)) and remembers the minimum of column \( j \) (denoted as \( ES_j = \min_{1 \leq i \leq m} \{D[i, j]\} \)) in step \( j \). For example, the first column minimum is 2 and accounts for the mappings of the 1-prefix of \( t \) to the two prefixes of \( s \) permitted by \( band_{1} \). As can be seen in the figure, the minima of the first few columns are only loose bounds and a pruning with Early Stopping alone is only possible in the very last step in this example.
Algorithm 13.1: APD(q, t, ε, pf, gd, band)

1. \(m1 = |q|; m2 = |t|;\)
2. \(\text{col} = \text{new double}[m1+1]; \quad // \text{define dummy column with } m1+1 \text{ entries}\)
3. \(\text{col}[1] = 0; \quad // \text{dummy row within dummy column}\)
4. \(\text{for } i=2 \text{ to } m1+1 \text{ do}\)
5. \(\quad \text{col}[i] = \infty;\)
6. \(\text{endfor}\)
7. \(\quad // \text{iterate over all columns of the cumulative distance matrix}\)
8. \(\text{for } j=1 \text{ to } m2 \text{ do}\)
9. \(\quad // \text{Fill a column with DTW components}\)
10. \(\quad \text{col} = \text{CalcDTWColumn}(q, t[j], \text{col}, \text{gd}, \text{band}[j]);\)
11. \(\quad \text{APD}_j = \min(\text{col}) + \text{pf}[m2+1-j]; \quad // \text{pf starts with } \text{pf}[1] = 0\)
12. \(\quad \text{if } \text{APD}_j > \epsilon \text{ then return } \infty; \quad // \text{pruning}\)
13. \(\text{endfor}\)
14. \(\quad // \text{return DTW value that is now stored in last entry of } \text{col}\)
15. \(\text{return } \text{col}[m1+1];\)

For Anticipatory Pruning, the filter piece \(pf_{10-1}\) is added. It lower-bounds all possible alignments for reversed time series starting in cell (10, 10) and ending in column 2. Thus, \(\text{APD}_1\) is obtained as “\(ES + pf^- = 2 + 18 = 20\)”, which is much closer to the DTW distance than the column minimum alone. Yet it is lower than the pruning threshold of 22. Continuing with filling the matrix, the second column minimum is 5 and the filter entry \(pf^- = 15\), yielding 20 again. In step 5, the column minimum is 13 and the corresponding filter distance piece is 10. The sum of 23 exceeds the pruning threshold 22. Hence, the DTW computation can be stopped. Thus, half of the ground distance computations are skipped and \(t\) is discarded.

The computation of APD for a query \(q\), a database time series \(t\), a pruning threshold \(\epsilon\), a piecewise filter \(pf\), a ground distance \(gd\), and a \textit{band} for the DTW is given by Algorithm 13.1. The computation of the column entries of the cumulative cost matrix is performed by Algorithm 13.2 as in the normal DTW case. Thus, the APD algorithm can return the final DTW distance without further calculations if no pruning was possible. The worst-case time complexity of APD is unchanged from the one of DTW. Compared with the according definitions, the algorithms use indices shifted by one for array \(pf\) (as arrays start at index 1.
Algorithm 13.2: CalcDTWColumn(q, tj, col, gd, bandj)

1. \( m_1 = |q| \);
2. // create a new column for the result; initialize the dummy row
3.   newcol[1] = \( \infty \);
4. // iterate over rows of the cumulative distance matrix; skip dummy row
5. for \( i = 2 \) to \( m_1 + 1 \) do
6.   if \( (i-1) \in \text{bandj} \) then
7.     // compute DTW entry; adjust for dummy row
8.       newcol[i] = gd(q[i-1], tj) + \min(newcol[i-1], col[i], col[i-1]);
9.   else
10.      newcol[i] = \( \infty \);
11. endif
12. endfor
13. return newcol;

here) and for the number of rows in a column (as the introduction of a guardian or dummy dimension simplifies the implementation).

Algorithm 13.3 shows how APD can be used for a range-based similarity search. Since APD returns either the DTW value or \( \infty \) in case of a pruning decision, the algorithm is virtually identical to a conventional multi-step DTW algorithm. The definition of the procedure calcPiecewiseFilter depends on the piecewise filter that is to be used (possible choices are discussed in Section 13.2.8). It returns an array of piecewise filter results instead of a single lower-bounding value. The only other necessary change is the call to APD in line 6 instead of the usual refinement call to a DTW procedure. Algorithms for \( k \)-nearest-neighbor searches and ranking queries can be adapted accordingly.

The following theorem states that \( \text{APD}_j \) is a lower bound for the DTW in each step \( j \). This property is central for the completeness of the result sets of similarity search algorithms based on APD.

**Theorem 13.3 (AP lower-bounds DTW)**

The Anticipatory Pruning Distance \( \text{APD}_j \) as given by Definition 13.4 is a lower bound of the DTW distance between two time series \( s \) and \( t \) of length \( m_1 \) and \( m_2 \).

\[
\forall 1 \leq j \leq m_2 : \text{APD}_j(s, t) = \min_{1 \leq i \leq m_1} \{ D[i, j] \} + \text{pf}_{m_2-j}(s^-, t^-) \leq \text{DTW}(s, t)
\]
Algorithm 13.3: aDTW-rangeQ(q, $\epsilon$, gd, band, DB)

1. ResultSet = $\emptyset$
2. for $t \in DB$ do
3.     m2 = $|t|$;
4.     pf = calcPiecewiseFilter(q, t, gd, band);
5.     // check if final result from filter allows pruning
6.     if pf[m2+1] $\leq$ $\epsilon$ then
7.         // check if APD allows pruning; returns $\infty$ if so; DTW otherwise
8.         if APD(q, t, $\epsilon$, pf, gd, band) $\leq$ $\epsilon$ then
9.             ResultSet = ResultSet $\cup$ {t};
10.        endif
11.    endif
12. endfor
13. return ResultSet;

Proof 13.3
The overall proof is based two properties of the DTW.

1. For all steps $j$, the column minimum is a lower bound of the path costs for paths ending in column $j$.

2. DTW is reversible.

The first property is trivially true due to $\min(A \cup \{t\}) \leq t$ for any finite set $A \subset \mathbb{R}$ and $t \in \mathbb{R}$. The second property was proven in in Section 13.2.6.

The remainder of the proof shows that these two properties together result in the lower-bounding property of APD as a series of partial warping path costs combined with a lower-bounding estimate of DTW values for time series postfixes.

Due to the continuity of DTW, a warping path $P = \{p_1, \ldots, p_l\}$ of minimal cost (cf. Definition 13.2) passes through all columns and thus also through both column $j$ and column $j + 1$ of the cumulative distance matrix $D$. Thus, the path can be decomposed into two parts $P_1 = \{p_1, \ldots, p_u\}$ and $P_2 = \{p_{u+1}, \ldots, p_l\}$ such that $P_1$ starts in the first column and ends in column $j$ while $P_2$ starts in column $j + 1$ and ends in the last column.

With $p_u = (x, j)$ and $p_{u+1} = (y, j+1)$ for some $1 \leq x, y \leq m_1$, the right-hand
Reducing the Number of DTW Ground Distance Computations

The side of the inequality can be rewritten as

\[ DTW(s, t) = D[x, j] + DTW(s[y : m1], t[(j + 1) : m2]). \]

From property 1 it follows that

\[ \min_{1 \leq i \leq m1} \{D[i, j]\} \leq D[x, j]. \]

From property 2 it follows that

\[ DTW(s[y : m1], t[(j + 1) : m2]) = DTW(s[m1 : y], t[m2 : (j + 1)]). \]

This term is underestimated by \( pf_{m-1}(s^-, t^-) \) by definition of the piecewise lower bound \( pf \). Together, the theorem follows.

Now that it has been shown that APD is a lower bound of DTW given some piecewise lower bound \( pf \), it immediately follows that Algorithm 13.3 is lossless and returns the same result as \( rangeQ(q, \epsilon, DTW_{gd,band}, DB) \) from page 20.

13.2.8 Piecewise Lower-Bounding Filters

In this section, existing piecewise lower-bounding filter techniques for DTW are discussed. The piecewise property is shown for the \( LB_{Keogh} \) lower bound of the DTW with details on other lower bounds in [AWK+09].

**Linearization**

The basic idea for linearization of the DTW computation for efficient and exact indexing is based on computing an envelope of upper and lower values around the query time series \( q \) of length \( m \) with respect to the \( k \)-band:

\[
UB_q[i] = \max_{i-k \leq j \leq i+k} \{q[j]\}
\]

\[
LB_q[i] = \min_{i-k \leq j \leq i+k} \{q[j]\}.
\]
The $\text{MinDist}$ of the ground distance between time series $t$ and the rectangle defined by vectors $UB_q$ and $LB_q$ lower-bounds the $k$-band DTW distance [Keo02]:

$$LB_{\text{Keogh}}(q, t) = \text{MinDist}_{L_2}(t, LB_q, UB_q) \leq DTW_{d_{L_2},k\text{-band}}(q, t)$$

with

$$\text{MinDist}_{L_2}(x, l, u) = \sqrt{\sum_{i=1}^{m} \begin{cases} (x[i] - u[i])^2 & x[i] > u[i] \\ (x[i] - l[i])^2 & x[i] < l[i] \\ 0 & \text{otherwise.} \end{cases}}$$

**Theorem 13.4**

$LB_{\text{Keogh}}$ is a piecewise lower-bounding filter as defined in Definition 13.3.

**Proof 13.4 (Sketch)**

As $LB_{\text{Keogh}}$ is a dimension-wise summation of distance components, the decomposition into a set of filters $pf_j$ for increasing sequence length $j$ is straightforward:

$$pf_j(q, t) = \sqrt{\sum_{i=1}^{j} \begin{cases} (t[i] - UB_q[i])^2 & t[i] > UB_q[i] \\ (t[i] - LB_q[i])^2 & t[i] < LB_q[i] \\ 0 & \text{otherwise.} \end{cases}}$$

As the sum takes the worst-case estimate that the warping path has one cell per column (there is only one entry in the sum per column) and that this cell contributes no more than the least possible ground distance amount from the column ($LB$ and $UB$ are the minimum/maximum over entries from $q$ according to $\text{band}_j$), it does not only lower-bound $DTW(q[1:j], t[1:j])$ but also $DTW(q[1:i], t[1:j])$ for $(i, j) \in \text{band}_j$.

In the experiments, Anticipatory Pruning is evaluated for $LB_{\text{Keogh}}$ with a tighter version of the envelope [ZS03] in combination with Piecewise Aggregate Approximation (PAA) of the time series.
Corner Boundaries

A generalization of the approach above can be found in [ZW07]. The generalized lower bounds are based on sets of non-overlapping regions or boundaries in the cumulative distance matrix through which any warping path has to cross. A sum over lower bounds of distance entries of the regions through which the warping path must pass results in a lower bound for the DTW. The authors propose several boundaries among which the hybrid approach \( LB_{Hybrid} \) has corner-shaped boundaries for the bottom left and top right of the matrix and straight vertical lines in the middle part of the matrix. The piecewise lower bound can be defined similar to the one above by summing over a subset of the boundaries.

Dimensionality Reduction and Indexing

Dimensionality reduction is a useful technique for efficiency gains in time series similarity search as many time series are considerably long (in this context the length of a time series is customarily interpreted as its dimensionality). Several approaches have been suggested. The aforementioned Piecewise Aggregate Approximation (PAA) used for instance in [Keo02] replaces parts of the original time series with constant approximations. As the name PAA suggests, the segments are piecewise by their very nature and fulfill the respective requirement of Anticipatory Pruning.

A number of DTW speed-up techniques also use indexing structures (e.g., R-trees in [Keo02] or sequential structures in [SYF05]). As Anticipatory Pruning is orthogonal to such techniques, using Anticipatory Pruning with index structures requires merely a change in the computation of the refinement step.

13.3 Experiments

13.3.1 Setup and Datasets

All experiments described in this section were executed on 2.33GHz Intel XEON CPU cores running JAVA implementations of the algorithms. The following default parameters were used where not stated otherwise: width of DTW band
13.3. Experiments

Figure 13.4: Sample of synthetic time series

$k = 40$, length of the time series $m = 512$, number of nearest neighbors retrieved peer query $= 10$, query workload per parameter combination $= 200$. Some additional experiments can be found in [AWK+09].

In addition to several real world datasets, two synthetic datasets $RW1$ and $RW2$ were generated for the experiments on the scalability in the number of attributes $n$ per point in time. Both synthetic datasets contain time series of length 512 and are of cardinality 10,000 with the number of attributes being varied between 1 and 50. All synthetic time series attributes were normalized to an average value of zero.

$RW1$: The non-normalized value of the $j^{th}$ attribute ($j \in \{1, \ldots, n\}$) of $t_{i+1}$ is a random walk value drawn from a normal distribution with parameters $\mu = 0$, $\sigma^2 = 1$ added to the value of the $j^{th}$ attribute of $t_i$: $t_{(i+1)}[j] = t_i[j] + N(0, 1)$. Univariate samples are shown in Figure 13.4(a).

$RW2$: The first two elements of the time series are generated as in $RW1$. For the remaining points in time of $RW2$, the average value $\mu$ depends on the last increase/decrease: $t_{(i+1)}[j] = t_i[j] + N(t_i[j] - t_{(i-1)}[j], 1)$. Examples are shown in Figure 13.4(b).

The three real world datasets are $SignLanguage$, $TRECvid$, and $NEWSVid$.

$SignLanguage$: This multivariate dataset is derived from the 11 real-valued attributes of the sign language finger tracking data from [Kad99]. For the efficiency experiments, the time series were created from the concatenated raw
Figure 13.5: Absolute efficiency improvement (average query time) on RW2 data by extracting non-overlapping windows of equal length. The length $m$ of the non-overlapping time series was varied between 64 and 512 and the band $k$ was varied between 10 and 150. The number of time series was fixed at 1,400.

**TRECVid & NEWSVid**: The two remaining datasets are derived from videos. The first one is based on the TRECVid benchmark data [SOK06]. The NEWSVid dataset consists of TV news shows that were recorded at 30 fps. 20-dimensional HSV histograms were computed for each video frame (i.e., the time series are multivariate with $n = 20$). The length $m$ of the non-overlapping video sequences was varied between 64 and 2048 frames. The cardinality of the database depends on the length of the time series and varies between 650 and 2,000 for TRECVid and between 2,000 and 8,000 for NEWSVid.

### 13.3.2 Results

The runtime performance and the number of ground distance calculations of aDTW are compared with those of the full DTW and DTW with Early Stopping. As aDTW depends on a piecewise filter, variants $LB_{Keogh}$ and $LB_{Hybrid}$ from Section 13.2.8 are investigated. The result sets computed in the experiments are identical for all approaches.

**Dimensionality Reduction**

In the first set of experiments, the efficiency of aDTW with respect to dimensionality reduction in the filter step is evaluated on the RW2 dataset. Figure 13.5
Anticipatory pruning, denoted as AP in the experiments, yields substantial run-time improvements compared with the base methods $LB_{Keogh}$ and $LB_{Hybrid}$. It also outperforms the Early Stopping approach (ES) [SYF05] for both filters. Anticipatory Pruning is especially helpful for strong reductions where it makes up for the loss in selectivity of the filter step by pruning during the refinement. Figure 13.6 summarizes the experiment by showing the relative improvement versus the respective filter-and-refine method with full DTW refinement computations. To abstract from implementation details, figures 13.7 to 13.11 demonstrate the reduction of the number of required ground distance calculations compared with the full DTW computation.

For the second synthetic dataset, a summary of the relative improvement is given by Figure 13.7(b). For this dataset of time series that rapidly oscillate around the mean value, the improvements are pronounced and less sensitive to the reduction parameter. The largest difference in the number of calculations can be seen for the $LB_{Hybrid}$ method, which shows a reduction of 75% (Anticipatory Pruning) versus 40% (Early Stopping). On the real world NEWSVid dataset, the behavior is quite similar to the one on RW1.
Univariate and Multivariate Time Series

The next set of experiments evaluates the effect of the number of attributes $n$ on Anticipatory Pruning. Based on the results shown in Figure 13.5, a reduction to 16 dimensions was chosen, which also represents an approximate upper limit for the indexing of time series in structures such as the R-Tree. As depicted in Figure 13.8(a) for the RW2 dataset, Anticipatory Pruning shows significant improvements for both filters. For higher numbers of attributes, the improvements of Anticipatory Pruning over Early Stopping increase with Early Stopping being unable to skip a significant number of computations for both the boundary and the linearization approach on the noisier dataset. On RW1, Anticipatory Pruning outperforms Early Stopping but both methods show good results for the univariate case, where a large difference in the starting value of two time series with high probability results in a large difference for the whole band of the first few columns.
Figure 13.9: Efficiency improvement (#calc.) for varying DTW bandwidths (Sakoe-Chiba band)

Figure 13.10: Efficiency improvement (#calc.) for varying time series lengths

**Bandwidth**

In Figure 13.9(a), the influence of the bandwidth constraint is studied on the NEWSVid dataset. The dimensionality reduction parameter was again set to 16 dimensions. A remarkable reduction in the number of calculations can be observed for the proposed Anticipatory Pruning based on both lower-bounding filters and for all evaluated bandwidths. The relative number of calculations avoided by Anticipatory Pruning is not very sensitive to the overall number of calculations that have to be performed. Even for extremely wide bands of 150 points in time, Anticipatory Pruning yields substantial relative improvements. The effect is also present for both the TRECVID dataset (Figure 13.9(b)) and the SignLanguage dataset (Figure 13.9(c)).
Reducing the Number of DTW Ground Distance Computations

Figure 13.11: Efficiency improvement (#calc.) for varying numbers of NN

Length of Time Series

For many applications, the length of the time series to be compared is of greater importance than the bandwidth of the DTW. In the experiment shown in Figure 13.10, the dimensionality reduction parameter in the filter was set to one quarter of the original time series length. Figure 13.10(a) and Figure 13.10(b) show improvements for the lower-bounding filters on the video datasets with remarkable improvements for Anticipatory Pruning when the length increased. The longer the time series is, the more distance information (in relative terms) does the anticipatory part contribute at the beginning of the APD computation. As Figure 13.10(c) shows, the results are similar on the SignLanguage dataset.

Number of Nearest Neighbors

As the last experiment, the effect of the number of nearest neighbors to be retrieved from the NEWSVid dataset are depicted in Figure 13.11. Neither Early Stopping nor Anticipatory Pruning are very sensitive to this parameter with a slight decrease being expected as a larger number of nearest neighbors results in a larger pruning distance.
13.4 Summary and Outlook

Existing techniques for speeding up DTW-based similarity searches aim at efficiently filtering the database and computing the full DTW in the refinement step. Unlike these algorithms, the aDTW approach proposed here exploits unused information from the filter step during the refinement and allows for a fast rejection of false candidates without having to calculate all ground distances of the cumulative distance matrix. As aDTW is a drop-in replacement for the DTW refinement distance, it can be integrated with other filter and indexing approaches whenever a piecewise filter is available. The significant reduction in the number of ground distance computations that aDTW achieves is robust with regard to many of the common parameters encountered when searching for similar time series. Together, these properties make the utilization of more expensive DTW ground distances viable as has been recently discussed for $\text{DTW}_{EMD}$ in [AK09, AKS10], where an efficient and effective video copy detection system is forged.
Part VI

Summary and Outlook
In distance-based similarity search, a field of research that is relevant to both data mining and information retrieval tasks, a distance measure quantifies a degree of similarity between two objects. In this context, a large distance value is interpreted as a small degree of similarity and vice versa. The effectiveness of the employed distance measures with regards to reflecting a desired notion of similarity and the efficiency of search algorithms based on these measures are of foremost importance for the overall system performance.

This work addressed a particularly intuitive and flexible class of distance measures. Transformation-based distance measures define a distance value between two objects by computing the least-cost transformation from a feature representation of one object to a feature representation of the other object. The cost of a single transformation in the feature space is modeled via a ground distance measure, which enables the incorporation of perceptual correlation across differing features. These two concepts make transformation-based distance measures computationally more expensive than many simple distance measures such as the Euclidean distance but they also endow them with a high degree of flexibility and a great effectiveness potential. Transformation-based distance measures thus constitute a sensible choice for a large number of similarity search tasks.

After having introduced the general concepts of distance-based similarity search in Part I, three prominent representatives of the class of transformation-based distance measures were reviewed in Part II: The Edit Distance, the Dynamic Time Warping distance, and the Earth Mover’s Distance, the latter of which was the focus of much of the rest of this work. Part III proposed two techniques that address the efficiency of the Earth Mover’s Distance. The first technique was based on a dimensionality reduction of the feature representations while the second technique explored direct and vector approximation-based indexing of the Earth Mover’s Distance. In Part IV, the effectiveness of the Earth Mover’s Distance as a measure of dissimilarity was addressed by giving a framework for its metric adaptation and showing how its flexibility in terms of the ground distance can be exploited in a Relevance Feedback setup. Finally, Part V explored new ways of applying transformation-based distances to objects that exhibit a structure such as graphs and time series. An application of the Earth
Mover’s Distance to graph similarity search was described based on a mapping from vertex degree co-occurrences to feature signatures, and the use of the Earth Mover’s Distance as a ground distance for Dynamic Time Warping was prepared by reducing the ground distance computations for the latter in a multi-step similarity search system. The results of empirical evaluations presented in this work show considerable improvement over existing techniques and/or baseline solutions.

Since 2005, the Earth Mover’s Distance as a transformation-based distance measure has been a prominent topic at the group of Professor Seidl at RWTH Aachen University. Parts of the research presented in this work were funded by grants SE1039/1-2 and SE1039/1-3 of the German Research Foundation DFG ending in 2010 (German title: “Schnelle inhaltsbasierte Suche in großen Multimedia-Datenbanksystemen mittels der Earth Mover’s Distance”). Additionally, the application of transformation-based distance functions to fingerprint-based authentication was explored as part of the joint BioKeyS project conducted in collaboration with the German Federal Office for Information Security (BSI), Fraunhofer IGD, Darmstadt University, and LMU Munich.

Beyond the potential for further research identified in chapters 8 to 13, the concept of feature signatures is being explored in the context of quadratic form distances in the group of Professor Seidl, and topics such as the application of the Earth Mover’s Distance and related distance measures to closed-loop control of combustion engines via the incorporation of higher moment information are envisioned to be explored as part of the second funding round of the DFG-funded special research field SFB686 at RWTH Aachen University starting in 2010.
Part VII

Appendices
Naming Conventions

Throughout the document, the following conventions were followed unless indicated otherwise for the sake of readability.

- **objects**: Database and query objects are denoted in lower case after \( n \) in the alphabet (e.g., \( o, p, q, r, s, t, u \)).

- **sets**: Sets start with an upper case letter (e.g., \( S, DB, Feasible \)).

- **vector and matrix access**: Individual dimensions of vectors and entries of matrices are accessed with square brackets (e.g., \( c[1], A[1, 2] \)).

- **set access**: Sorted set elements are accessed with square brackets (e.g., \( DB[1], KNN[k] \)).

- **range access**: Ranges are given by a colon (e.g., \( DB[k : 2k], c[1 : n'] \)).

- **enumeration**: Numbered objects have a lower index (e.g., \( o_1, o_2 \)).

- **association**: General association of an object with another object is denoted by an upper or lower index (e.g., \( c^p, EMD_{ed} \)).

For example, \( c^p_2[1 : n'] \) is the second vector \( c \) associated with set \( P \) projected on the first \( n' \) dimensions while \( DB[5][3] \) is the value of the third dimension of the \( 5^{th} \) object in \( DB \) where set \( DB \) is accessed as an ordered set of vectors/arrays.
# List of Symbols

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$q$</td>
<td>query object</td>
</tr>
<tr>
<td>$o, p, r, s, t, u$</td>
<td>database objects</td>
</tr>
<tr>
<td>$h^o$</td>
<td>fixed-binning feature histogram of object $o$ (p. 10)</td>
</tr>
<tr>
<td>$s^o$</td>
<td>adaptive-binning feature histogram/signature of object $o$ (p. 11)</td>
</tr>
<tr>
<td>$\langle t_0^o, \ldots, t_m^o \rangle$</td>
<td>feature sequence of object $o$ (p. 13)</td>
</tr>
<tr>
<td>$t^-$</td>
<td>reversed sequence/time series $\langle t_m^o, \ldots, t_1^o \rangle$ (p. 225)</td>
</tr>
<tr>
<td>$</td>
<td>o</td>
</tr>
<tr>
<td>$HS^{p</td>
<td>r}$</td>
</tr>
<tr>
<td>$VC^{p,S}$</td>
<td>Voronoi cell of point $p$ in a space $S$ (p. 35)</td>
</tr>
<tr>
<td>$VD^{DB,S}$</td>
<td>Voronoi diagram of point set DB in a space $S$ (p. 35)</td>
</tr>
<tr>
<td>$\chi$</td>
<td>a characteristic filter function (p. 46)</td>
</tr>
<tr>
<td>$\mathbb{R}^{n,n'}$</td>
<td>the set of combining linear reduction matrices (p. 99)</td>
</tr>
<tr>
<td>$A \cdot B$</td>
<td>the element-wise matrix multiplication (p. 157)</td>
</tr>
<tr>
<td>$(A \cdot a_{ij})$</td>
<td>single-element matrix multiplication (p. 157)</td>
</tr>
<tr>
<td>$\mathbb{R}$</td>
<td>the set of relevant objects in a database (p. 175)</td>
</tr>
<tr>
<td>$\delta^G(v)$</td>
<td>the edge degree of vertex $v$ in a graph $G$ (p. 204)</td>
</tr>
<tr>
<td>$P^g_m$</td>
<td>the set of all simple paths of length $m$ in $G$ (p. 205)</td>
</tr>
<tr>
<td>$D^g_m$</td>
<td>the vertex degree co-occurrence multiset of degree $m$ for graph $G$ (p. 206)</td>
</tr>
<tr>
<td>$\mathbb{R}, \mathbb{R}^+$</td>
<td>the set of real numbers and the set of non-negative real numbers</td>
</tr>
<tr>
<td>$\mathbb{N}_0, \mathbb{N}_1$</td>
<td>the set of natural numbers starting with 0 and 1</td>
</tr>
<tr>
<td>$\mathbb{Z}$</td>
<td>the set of integers</td>
</tr>
</tbody>
</table>
# List of Algorithms

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.1</td>
<td>rangeQ(q, ε, d, DB)</td>
<td>20</td>
</tr>
<tr>
<td>2.2</td>
<td>nnQ(q, k, d, DB)</td>
<td>22</td>
</tr>
<tr>
<td>2.3</td>
<td>rankingQ(q, i, k, d, DB)</td>
<td>23</td>
</tr>
<tr>
<td>3.1</td>
<td>rtree-rangeQ(q, ε, d, minDist, RTree)</td>
<td>29</td>
</tr>
<tr>
<td>3.2</td>
<td>rec-rtree-rangeQ(q, ε, d, minDist, node)</td>
<td>29</td>
</tr>
<tr>
<td>4.1</td>
<td>voronoi-nnQ(q, VD_{DB, S})</td>
<td>35</td>
</tr>
<tr>
<td>4.2</td>
<td>approx-voronoi-nnQ(q, d, AVD_{DB, S})</td>
<td>36</td>
</tr>
<tr>
<td>5.1</td>
<td>editDistance(s,t,gd)</td>
<td>64</td>
</tr>
<tr>
<td>8.1</td>
<td>optimizeFB_MOD(R, C, F, n, n')</td>
<td>112</td>
</tr>
<tr>
<td>8.2</td>
<td>calcQuality(R, C, F, origDim, newRedDim, n')</td>
<td>113</td>
</tr>
<tr>
<td>8.3</td>
<td>optimizeFB_ALL(R, C, F, d, d')</td>
<td>114</td>
</tr>
<tr>
<td>9.1</td>
<td>computeLT(C, q, ColSorting, LBoundaries)</td>
<td>138</td>
</tr>
<tr>
<td>9.2</td>
<td>computeLBLT(approx_o, LT)</td>
<td>139</td>
</tr>
<tr>
<td>9.3</td>
<td>LB_LT-rangeQ(q, ε, EMD_{C}, DB, ColSorting, VA)</td>
<td>139</td>
</tr>
<tr>
<td>9.4</td>
<td>LB_LT^-rangeQ(q, ε, EMD_{C}, DB, ColSorting, EntrySorting, VA)</td>
<td>142</td>
</tr>
<tr>
<td>11.1</td>
<td>feedbackLoop(DB, q, gd, k, computeQuery, computeGD)</td>
<td>173</td>
</tr>
<tr>
<td>11.2</td>
<td>optimize(C, q, feedback)</td>
<td>177</td>
</tr>
<tr>
<td>11.3</td>
<td>computeGD-QM(q, feedback, iter)</td>
<td>179</td>
</tr>
<tr>
<td>11.4</td>
<td>computeGD-Histogram-Heur(q, feedback, iter)</td>
<td>180</td>
</tr>
<tr>
<td>11.5</td>
<td>computeGD-Histogram-Heur-Matrix(q, feedback, iter)</td>
<td>180</td>
</tr>
<tr>
<td>11.6</td>
<td>computeGD-Histogram-Opt(q, feedback, iter)</td>
<td>182</td>
</tr>
<tr>
<td>Algorithm Description</td>
<td>Page</td>
<td></td>
</tr>
<tr>
<td>-----------------------</td>
<td>------</td>
<td></td>
</tr>
<tr>
<td>11.7 modify(C)</td>
<td>183</td>
<td></td>
</tr>
<tr>
<td>11.8 computeGD-Signature-Heur(q, feedback, iter)</td>
<td>184</td>
<td></td>
</tr>
<tr>
<td>11.9 computeGD-Signature-Heur-Matrix(q, feedback, iter)</td>
<td>185</td>
<td></td>
</tr>
<tr>
<td>11.10 computeGD-Signature-Opt(q, feedback, iter)</td>
<td>187</td>
<td></td>
</tr>
<tr>
<td>13.1 APD(q, t, ε, pf, gd, band)</td>
<td>227</td>
<td></td>
</tr>
<tr>
<td>13.2 CalcDTWColumn(q, tj, col, gd, bandj)</td>
<td>228</td>
<td></td>
</tr>
<tr>
<td>13.3 aDTW-rangeQ(q, ε, gd, band, DB)</td>
<td>229</td>
<td></td>
</tr>
</tbody>
</table>


Norbert Beckmann and Bernhard Seeger. A Revised R*-Tree in Comparison with Related Index Structures. In *Proceedings of the
ACM International Conference on Management of Data (SIGMOD),


pages 204–221, 2006.


[CFY03] Franky K.-P. Chan, Ada W.-C. Fu, and Clement Yu. Haar wavelets for efficient similarity search of time-series: With and without


Collaboration / Acknowledgments

This thesis would not have been possible without the close collaboration in the group of Professor Seidl. Many of the presented ideas and techniques evolved from numerous fruitful discussions in the group. The close-contact style of advising students encouraged by Professor Seidl allowed many of my ideas to be evaluated, scrapped, rethought, refined, reevaluated and sometimes published in collaboration with my advisees and colleagues. While the high level of collaboration both in the group and with the students constitutes a key factor for the productive environment I found at Professor Seidl's group, it makes it hard to pinpoint individual contributions. I will try to clarify my contributions to the publications that resulted from the work presented in this thesis in order to give the outside reader a more complete picture than can be obtained by looking at the order of authors on the publications. While not listed explicitly for every publication, Professor Seidl is clearly to be credited with making numerous suggestions for improvements and additions that ultimately helped get many of the concepts and techniques I worked on accepted for publication in the proceedings of major database conferences.

The work I did while being advised by Christoph Brochhaus served as the foundation for the paper on Voronoi-based nearest neighbor processing published in the proceedings of EDBT. Its general framework was developed in cooperation with Christoph, who also helped with the selection of a subset of techniques to be included in the paper, with the choice of experiments to be presented, and with part of the formulation of the text. I devised the core concepts and formalized, implemented, and evaluated them for the EDBT paper. Since this paper was my first publication at a major conference, Christoph's strategic advice on how to formulate and present techniques for submission and
ultimately acceptance was most helpful.

The dimensionality reduction for the Earth Mover’s Distance described in Chapter 8 is based on work done in close collaboration with Philipp Kranen, whom I tasked with looking at differing reductions for his diploma thesis. In his thesis that Ira Assent helped co-advising, Philipp went to great length to identify which reduction techniques had potential and which did not. This allowed me to focus on describing, implementing, and evaluating linear reductions with sparse reduction matrices for the paper published in the proceedings of SIGMOD. Both Ira Assent and student-turned-colleague Philipp Kranen were involved with the structuring and the formulation of the paper. Ira in particular helped with the description of the general EMD model and Philipp worked on the reduction-specific text together with me.

In the ICDE publication on indexing techniques for the Earth Mover’s Distance on which Chapter 9 is based, my work was focused on the direct indexing part, the MinDist, its lower-bounding proof, and its experimental evaluation (experiment design & implementation). The ideas presented in the part on the VA-File are to be largely credited to Ira Assent and Tobias Meisen; here, I helped putting them into writing.

The framework for metric adaptation of the EMD presented in Chapter 10 is based on a German publication in the proceedings of the BTW conference. The general concept spun off in the course of advising Christian Beecks on his diploma thesis. Christian and I later revised the concept for publication. Additional formalizations, a discussion on the EMD iso-surface, and further proofs that I worked on and go beyond the published paper are included in Chapter 10.

The ideas for optimization-based relevance feedback for the EMD was developed while advising Martin Sundermeyer on his diploma thesis. In addition to describing, formalizing, and structuring the ideas, the evaluation (and the underlying implementation) were my doing. Christian Beecks helped with the planning of papers that started during at the 2008 Dagstuhl school on information retrieval and with the formulation of parts of both the paper published in the proceedings of CIKM and the paper accepted for the AMR workshop. Martin was available for proof-reading both papers. Chapter 11 includes additional techniques on fixed-binning histograms not included in the two publications.
Credit to part of the graph-related implementation for the evaluations in
Chapter 12 goes to Anca Ivanescu, whom I advised during her diploma thesis
on the topic of graph similarity, and to student assistant Marcel Rosenthal. In
this work, I described my idea that was the outset for Anca's thesis.

The idea of the lower bound for the DTW (cf. Chapter 13) was largely
devised by Ira Assent, Ralph Krieger, and their advisee Hardy Kremer. I joined
the team of authors at a later stage and worked on the proofs, on the line of
reasoning, on the way of presenting the ideas in the paper, and on the design
and choice of experiments to be included. I additionally collaborated with Hardy
Kremer on the main post-submission work including the author feedback round
that turned out to be crucial for the publication in the proceedings of VLDB and
on the final rewriting that put some of the concepts and proofs more precisely
and concisely.
List of Publications


Curriculum Vitae

Name
Marc Wichterich

Academic Degree
Diplom-Informatiker

Born on
September 16th, 1978

Born in
Haan, Germany

Nationality
German

Education

from 05/2005
RWTH Aachen University, Germany
Computer science: Doctoral Studies

10/1999 – 04/2005
RWTH Aachen University, Germany
Computer science: Diplom Informatik

Hochdahl Secondary School (Gymnasium), Germany

Professional Experience

from 10/2005
Computer Science 9, RWTH Aachen, Germany
Research assistant

05/2005 – 09/2005
Computer Science 9, RWTH Aachen, Germany
Graduate assistant

11/2001 – 01/2002
Eonyt Unternehmensberatung GmbH, Germany
Project work

11/1999 – 03/2000
Parsytec AG, Germany
Student assistant

07/1999 – 08/1999
Parsytec Computer GmbH, Germany
Internship

07/1998 – 07/1999
ASB Düsseldorf, Germany
Mandatory civilian service in lieu of military service
“I'm sciencing as fast as I can!”  Professor H. Farnsworth [GCK07]