ORCHID – Reduction-Ratio-Optimal Computation of Geo-spatial Distances for Link Discovery

Axel-Cyrille Ngonga Ngomo

Department of Computer Science
University of Leipzig
Johannisgasse 26, 04103 Leipzig
ngonga@informatik.uni-leipzig.de
http://limes.sf.net

Abstract. The discovery of links between resources within knowledge bases is of crucial importance to realize the vision of the Semantic Web. Addressing this task is especially challenging when dealing with geo-spatial datasets due to their sheer size and the potential complexity of single geo-spatial objects. Yet, so far, little attention has been paid to the characteristics of geo-spatial data within the context of link discovery. In this paper, we address this gap by presenting ORCHID, a reduction-ratio-optimal link discovery approach designed especially for geo-spatial data. ORCHID relies on a combination of the Hausdorff and orthodromic metrics to compute the distance between geo-spatial objects. We first present two novel approaches for the efficient computation of Hausdorff distances. Then, we present the space tiling approach implemented by ORCHID and prove that it is optimal with respect to the reduction ratio that it can achieve. The evaluation of our approaches is carried out on three real datasets of different size and complexity. Our results suggest that our approaches to the computation of Hausdorff distances require two orders of magnitude less orthodromic distances computations to compare geographical data. Moreover, they require two orders of magnitude less time than a naive approach to achieve this goal. Finally, our results indicate that ORCHID scales to large datasets while outperforming the state of the art significantly.

Keywords: Link discovery, Record Linkage, Deduplication, Geo-Spatial Data, Hausdorff Distances.

1 Introduction

The Linked Open Data Cloud (LOD Cloud) has developed to a compendium of approximately 300 datasets over the last few years. Currently, geographic data sets contain approximately 6 billion triples and make up 19.4% of the triples in the LOD Cloud. Projects such as LinkedGeoData[1] promise an increase of these numbers by orders of magnitude in the near future. However, only 7.1% of the links between knowledge bases in the LOD Cloud currently connect geographic entities. This means that less than 1% of triples within the geographic datasets of the LOD Cloud are links between

[1] See http://linkedgeodata.org. Last access: January 11th, 2013.

H. Alani et al. (Eds.): ISWC 2013, Part I, LNCS 8218, pp. 395–410, 2013.

knowledge bases.[2] This blatant lack of links is partly due to two factors: First, it is due to the *large number of geo-spatial entities* available on the Linked Open Data Cloud. Moreover, the *geo-spatial resources* are often described as (often ordered) sets of points which describe geometric objects such as (multi-) polygons or (multi-) polylines. This way of describing resources differs considerably from the approach followed for most Linked Data resources, which are commonly easiest identified by the means of a label. Consequently, such descriptions have not yet been payed much attention to in the field of link discovery (LD).

We address this gap by presenting ORCHID, a reduction-ratio-optimal approach for LD. ORCHID assumes the LD problem as being formulated in the following way: Given a set S of source instances, a set T of target instances and a distance threshold θ, find the set of triples $(s, t, \delta(s, t)) \in S \times T \times \mathbb{R}^+$ such that $\delta(s, t) \leq \theta$. Given this assumption, the idea behind ORCHID is to address the LD problem on geographic data described as (ordered) sets of points by two means. First, ORCHID implements time-efficient algorithms for computing whether the distance between two polygons s and t is less or equal to a given distance threshold θ. Moreover, ORCHID implements a space tiling algorithm for orthodromic spaces which allows discarding yet another large number of unnecessary computations.

The rest of this paper is structured as follows: In Section 2, we present the core notation used throughout this paper as well as some formal considerations underlying our approach. Section 3 presents two approaches that allow computing the Hausdorff distance between two polygons efficiently.[3] Subsequently, we present the space discretization approach implemented by ORCHID and show that it is optimal with respect to its reduction ratio. We then present a thorough evaluation of our approach on three datasets of different sizes and complexity. We also compare our approach with a state-of-the-art LD framework which implements the orthodromic distance. We conclude the paper with a brief overview of related work (Section 6) and a discussion of our results (Section 7). The approach presented here was integrated in the LIMES framework.[4] Due to space restrictions, we had to omit some details of the approaches presented herein. These can be found in the corresponding technical report on the project webpage.

2 Preliminaries

The formal specification of LD adopted herein is tantamount to the definition proposed in [11]: Given a set S of source resources, a set T of target resources and a relation R, our goal is to find the set $M \subseteq S \times T$ of pairs $(s, t) \in S \times T$ such that $R(s, t)$. If R is `owl:sameAs`, then we are faced with a *deduplication task*. Given that the explicit computation of M is usually a very complex endeavor, M is usually approximated by a set $\tilde{M} = \{(s, t, \delta(s, t)) \in S \times T \times \mathbb{R}^+ : \delta(s, t) \leq \theta\}$, where δ is a distance function and $\theta \geq 0$ is a distance threshold. For geographic data, the resources s and t are described

[2] See `http://wifo5-03.informatik.uni-mannheim.de/lodcloud/state/` for an overview of the current state of the Cloud. Last access: January 11th, 2013.

[3] The Hausdorff distance can be used to compare the distance between any two sets of ordered points located in a space where a distance function is defined. Thus, while we focus on polygons in this paper, our approach can be used for all sets of points.

[4] `http://limes.sf.net`

by using single points or (ordered) sets of points, which we regard as polygons. Given that we can regard points as polygons with one node, we will speak of resources being described as polygons throughout this paper. We will use a subscript notation to label the nodes that make up resources. For example, if s had three nodes, we would denote them s_1, s_2, and s_3. For convenience's sake, we will write $s = \{s_1, s_2, s_3\}$ and $s_i \in s$.

While there are several approaches for computing the distance between two polygons [2], a common approach is the use of the Hausdorff distance [14] *hd*:

$$hd(s,t) = \max_{s_i \in s}\{\min_{t_j \in t}\{\delta(s_i,t_j)\}\}, \tag{1}$$

where δ is the metric associated to the affine space within which the polygons are defined. We assume that the earth is a perfect ball with radius $R = 6378\,km$. Then, δ is the *orthodromic distance* and will be denoted *od* in the rest of this paper. Given these premises, the LD task we investigate in this paper is the following: Find the set $\tilde{M} = \{(s,t,hd(s,t)) \in S \times T : hd(s,t) \leq \theta\}$ where $\forall s_i \in s\ \forall t_j \in t\ \delta(s_i,t_j) = od(s_i,t_j)$. It is important to notice that the orthodromic distance is known to be a metric, leading to the problem formulated above being expressed in a metric space.

Two requirements are central for the approaches developed herein. First, the approaches have to be *complete* (also called lossless [11]), which simply means that they must be able to compute *all triples* $(s,t,hd(s,t)) \in S \times T \times \mathbb{R}^+$ for which $hd(s,t) \leq \theta$ holds. This characteristic is not fulfilled by certain blocking approaches, which trade runtime efficiency for completeness. In addition to developing a complete approach, we aim to develop a reduction-ratio-optimal approach [10]: Let $\mathcal{A}$ be an algorithm for computing $\tilde{M}$ and α be the vector that contains all parameters necessary to run $\mathcal{A}$. Moreover, let $|A(\alpha)|$ be the number of computations of *hd* carried out by $\mathcal{A}$ when assigned the vector of parameters α. We call $\mathcal{A}(\alpha)$ reduction-ratio-optimal when

$$\forall r < 1 - \frac{|\tilde{M}|}{|S||T|}\ \exists \alpha : 1 - \frac{|A(\alpha)|}{|S||T|} \geq r. \tag{2}$$

Naive approaches to computing $\tilde{M}$ have two drawbacks: First, they require $|s||t|$ calls of *od* to compute $hd(s,t)$. Moreover, they carry out $|S||T|$ computations of *hd* to find all elements of $\tilde{M}$. Addressing the time complexity of LD on geographic data thus requires addressing these two quadratically complex problems. Our approach addresses the time complexity of the first problem by making use of the Cauchy-Schwarz inequality, i.e.,

$$od(x,y) \leq od(x,z) + od(z,y), \tag{3}$$

and of bounding circles for approximating the distance between polygons. The second problem is addressed by the means of a reduction-ratio-optimal tiling approach similar to the $\mathcal{HR}^3$ algorithm [10].

3 Efficient Computation of Hausdorff Distances

Several approaches have addressed the time-efficient computation of Hausdorff distances throughout literature (see [14] for a good overview). Yet, so far, these approaches

have not been concerned with the problem of only finding those triples $(s, t, hd(s, t))$ with $hd(s, t) \leq \theta$. In the following, we present several approaches for achieving this goal. These approaches are later evaluated in Section 5. For space reasons, we omit the pseudo-code for the first two approaches. These can be found in the technical report.

3.1 Naive Approach

The naive approach for computing $hd(s, t)$ would compare all elements of the polygon $s \in S$ with all elements of the polygons $t \in T$ by computing the orthodromic distance between all $s_i \in s$ and $t_j \in t$. Let $\bar{S}$ be the average size of the polygons in S and $\bar{T}$ be the average size of the polygons in T. The best- and worst-case runtime complexities of the naive approach are then $O(|S||T|\bar{S}\bar{T})$.

3.2 Bound Approach

A first idea to make use of the bound $hd(s, t) \leq \theta$ on distances lies in the observation that

$$\exists s_i \in S : \min_{t_j \in t}\{od(s_i, t_j)\} > \theta \rightarrow hd(s, t) > \theta \tag{4}$$

This insight allows terminating computations that would not lead to pairs for which $hd(s, t) \leq \theta$ by terminating the computation as soon as a s_i is found that fulfills Eq. (4). In the best case, only one point of each $s \in S$ is compared to all points of $t \in T$ before the computation of $hd(s, t)$ is terminated. Thus, the best-case complexity of the approach is $O(|S||T|\bar{T})$. In the worst case (i.e., in the case that the set of mappings returned is exactly $S \times T$), the complexity of the bound approach is the same as that of the naive approach, i.e., $O(|S||T|\bar{S}\bar{T})$.

3.3 Indexed Approach

The indexed approach combines the intuition behind the bound approach with geometrical characteristics of the Hausdorff distance by using two intuitions. The first intuition is that if the minimal distance between any point of s and any point of t is larger than θ, then $hd(s, t) > \theta$ must hold. Our second intuition makes use of the triangle inequality to approximate the distances $od(s_i, t_k)$. In the following, we present these two intuitions formally. We dub the indexed approach which relies on the second intuition alone *CS* while we call the indexed approach that relies on both intuitions *BC + CS*.

Intuition 1: Bounding Circles. Formally, the first intuition can be expressed as follows:

$$\min_{s_i \in s,\ t_j \in t}\{od(s_i, t_j)\} > \theta \rightarrow hd(s, t) > \theta. \tag{5}$$

Finding the two points s_i and t_j which minimize the value of $od(s_i, t_j)$ requires $O(|s||t|)$ computations of od, i.e., $O(|S||T|\bar{S}\bar{T})$ overall. However, a lower bound for this minimum for all pairs $(s, t) \in S \times T$ can be computed efficiently by using encompassing circles: Let $C(s)$ resp. $C(t)$ be the smallest circles that fully encompass s resp. t. Moreover, let

$r(s)$ resp. $r(t)$ be the radius of these circles and $\zeta(s)$ resp. $\zeta(t)$ be the centers of the circles $C(s)$ resp. $C(t)$. Then,

$$\min_{s_i \in s, t_j \in t} \{od(s_i, t_j)\} > od(\zeta(s), \zeta(t)) - (r(s) + r(t)) = \mu(s, t). \tag{6}$$

Figure 1 displays the intuition behind this approximation graphically. Note that this equation also holds when the circles overlap (in which case $od(\zeta(s), \zeta(t)) - (r(s) + r(t)) < 0$ as $od(\zeta(s), \zeta(t)) < (r(s) + r(t))$.

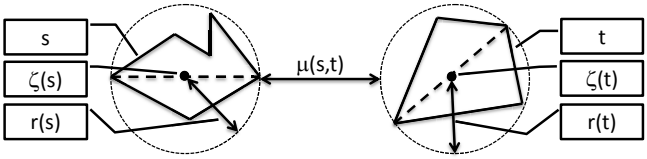


Fig. 1. Lower bound of Hausdorff distances based on circles

Computing the smallest circle that encompasses any polygon x can be carried out in $O(|x|^2)$ by simply computing $od(x_i, x_k)$ for all $(x_i, x_k) \in x^2$. Then,

$$r(x) = \frac{\max_{x_i \in x, x_k \in x} od(x_i, x_k)}{2} \tag{7}$$

while

$$\zeta(x) = \frac{x^+ + x^-}{2} \text{ where } (x^+, x^-) = \arg\max_{x_i \in x, x_k \in x} od(x_i, x_k). \tag{8}$$

The proof that the radius $r(x)$ must have the value shown in Equation 7 is as follows: The points within a circle with radius r' are at most at a distance $2r'$ of each other. Consequently, any circle with radius $r' < r(x)$ cannot contain both elements of the pair $(x^+, x^-) = \arg\max_{x_i \in x, x_k \in x} od(x_i, x_k)$. Thus, the smallest possible radius of a circle that encompasses x fully must be the maximal distance between points which belong to x. This is exactly the value of $r(x)$. Now the only way to ensure that a circle with radius $r(x)$ really encompasses all points in x is to have x^+ and x^- to be diametrically opposite. Thus, $\zeta(x)$ must be exactly in the middle of x^+ and x^-.

The runtime complexity of this approximation is $O(|S|\bar{S}^2 + |T|\bar{T}^2 + |S||T|)$. $O(|S|\bar{S}^2 + |T|\bar{T}^2)$ computations of od are required to determine the circles and their radii while $O(|S||T|)$ computations are required to compare the circles computed out of S with those from T. Note that for large problem $\bar{S}^2$ resp. $\bar{T}^2$ are very small compared to $|S|$ resp. $|T|$, leading to $O(|S|\bar{S}^2 + |T|\bar{T}^2 + |S||T|) \approx O(|S| + |T| + |S||T|) \approx O(|S||T|)$.

Intuition 2: Distance Approximation Using the Cauchy-Schwarz Inequality. Now given that we have computed all distances between all pairs $(t_j, t_k) \in t^2$, we can reuse this information to approximate distances from any s_i to any t_k by relying on the Cauchy-Schwarz inequality in a fashion similar to the LIMES algorithm presented

in [12]. The idea here is that we can compute an upper and a lower bound for the distance $od(s_i, t_k)$ by using the distance $od(s_i, t_j)$ previously computed as follows:

$$|od(s_i, t_j) - od(t_j, t_k)| \leq od(s_i, t_k) \leq od(s_i, t_j) + od(t_j, t_k). \quad (9)$$

For each s_i, exploiting these pre-computed distances can be carried out as follows: For all t_k for which $od(s_i, t_k)$ is unknown, we approximate the distance from s_i to t_k by finding a point t_j for which

$$t_j = \arg\min_{t_x \in t'} od(t_x, t_k) \quad (10)$$

holds, where $t' \subseteq t$ is the set of points t_x of t for which $od(s_i, t_x)$ is known. We call the point t_j an *exemplar* for t_k. The idea behind using one of points closest to t_k is that it gives us the best possible lower bound $|od(s_i, t_j) - od(t_j, t_k)|$ for the distance $od(s_i, t_k)$. Now if $|od(s_i, t_j) - od(t_j, t_k)| > \theta$, then we can discard the computation of the distance $od(s_i, t_k)$ and simply assign it any value $\Theta > \theta$. Moreover, if $|od(s_i, t_j) - od(t_j, t_k)|$ is larger than the current known minimal distance between s_i and points in t, then we can also discard the computation of $od(s_i, t_k)$. If such an exemplar does not exist or if our approximations fail to discard the computation, then only do we compute the real value of the distance $od(s_i, t_k)$.

The best-case complexity of this step alone would be $O(|S||T|\bar{S})$ while in the worst case, we would need to carry out $O(|S||T|\bar{S}\bar{T})$ computations of od. The overall complexity of the indexed approach is $O(|S|\bar{S}^2 + |T|\bar{T}^2 + |S||T|)$ (i.e., that of the bounding circles filter) in the best case and $O(|S|\bar{S}^2 + |T|\bar{T}^2 + |S||T| + |S||T|\bar{S}\bar{T})$ in the worst case. The overall algorithm underlying the indexed approach is shown in Algorithm 1.

4 ORCHID

Although the indexed method presented above can significantly reduce the number of computations carried out to compare S and T, it still needs at least $|S||T|$ comparisons. For example, imagine our source and target data sets were all geo-spatial entities on the portion of the surface of the planet shown in Figure 2. If Oslo (which has the coordinates (59?56'58" N, 10?45'23" E)) was the resource to link via `dbp:near`, then the approaches above would compare it with each of the other elements of the dataset. The idea behind Orchid is to reduce the number of comparisons even further while remaining complete and being reduction-ratio-optimal. To achieve this goal, Orchid uses a space discretization approach and only compares polygons $t \in T$ which lie within a certain range of $s \in S$. An example of the discretization generated by Orchid is shown in Figure 2. Instead of comparing Oslo with all other elements of the dataset, Orchid would only compare it with the geo-spatial objects shown in the gray cells. In the following, we present Orchid formally and prove that it is both complete and reduction-ratio-optimal.

4.1 Preliminaries

Explaining the approach implemented by Orchid prerequisites the explication of a set of characteristics of the orthodromic distance od. Given a polygon s, finding all points

Algorithm 1. Implementation of the $BC + CS$ Hausdorff distance computation. The implementation of CS lacks lines 1,2,3 and 28.

```
 1: if (od(c(s), c(t)) − r(s) − r(t) > θ) then
 2:     return ∅
 3: else
 4:     max ← 0
 5:     for s_i ∈ s do
 6:         min ← ∞
 7:         for t_j ∈ t do
 8:             e = exemplar(t_j)
 9:             if e ≠ ∅ then
10:                 approx = |od(s_i, e) − od(e, t_j)|
11:                 if approx > θ ∨ approx > min then
12:                     d(s_i, t_j) = θ + 1
13:                 else
14:                     d(s_i, t_j) = od(s_i, t_j)
15:                 end if
16:             else
17:                 d(s_i, t_j) = od(s_i, t_j)
18:             end if
19:             min = min(min, d(s_i, t_j))
20:         end for
21:         max = max(max, min)
22:     end for
23:     if max > θ then
24:         return ∅
25:     else
26:         return max
27:     end if
28: end if
```

t such that $hd(s,t) \leq \theta$ requires being able to find all points y for which $od(x,y) \leq \theta$ given a point x. In general, a point x on the surface of the planet can be characterized by two values: its latitude $lat(x)$ and its longitude $long(x)$. These two values are bound as $-\pi/2 \leq lat(x) \leq \pi/2$ and $-\pi \leq lon(x) \leq \pi$ always hold.[5] We write $x = (lat(x), lon(x))$ to denote points. Now, given a point y with $lon(x) = lon(y)$, then $od(x,y) = R|lat(x) - lat(y)|$. Yet, if $lat(x) = lat(y)$, then $od(x,y) = R|lon(x) - lon(y)|cos(lat(x))$. This difference between latitude and longitude is central when finding all points y for which $od(x,y) \leq \theta$. Formally, it means that we can create a discretization in which we treat the latitude values independently from the longitude values but not the other way around. This particular characteristic of latitude and longitude values lies at the heart of Orchid.

4.2 Discretization for Geo-Spatial Points

The idea behind Orchid is to make use of the values of latitude and longitude being bound to first create a grid on the surface of the planet. We call $\alpha \in \mathbb{N}$ the granularity

[5] All angles in this paper are assumed to be in radian unless stated otherwise.

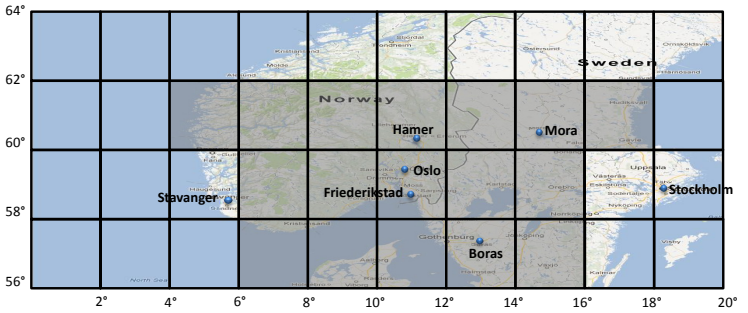


Fig. 2. Example of tiling for $\alpha = 1$ and $\theta = 222.6km$ (i.e., $\Delta_R = 2°$). Here, the resource to link is Oslo. The gray cells are the elements of A(Oslo).

parameter of Orchid. Given the premises described in Section 4.1, we can infer that for $x \in s \in S$ and $t \in T \in T$

$$od(x, y) \leq \theta \rightarrow |lat(x) - lat(y)| \leq \theta/R = \theta_R. \tag{11}$$

Based on this equation, we can create a grid such that width and height of each cell of the grid is $\Delta_R = \theta_R/\alpha$. For each cell c_i, two whole numbers c_i^{lat} and c_i^{lon} exist such that c_i contains only points x for which

$$(c_i^{lat}\Delta \leq lat(x) < (c_i^{lat} + 1)\Delta) \wedge (c_i^{lon}\Delta \leq lon(x) < (c_i^{lon} + 1)\Delta) \tag{12}$$

holds. We call (c_i^{lat}, c_i^{lon}) the coordinates of c_i. Moreover, we write $x \in c_i$ if Equation 12 holds for x. We also write $c_i(x)$ to signify the cell to which x belongs. In our example (Figure 2), the cell which contains Oslo has the coordinates (29, 5). Given this definition of a grid, the set $\tilde{M}(x)$ of y with $od(x, y) \leq \theta$ is clearly a subset of all y for which $|lat(x) - lat(y)| \leq \theta_R$ holds. With respect to our grid, we can infer the following inequality:

$$x \in c_i \wedge y \in c_j \wedge |c_i^{lat} - c_j^{lat}| > \alpha \rightarrow y \notin \tilde{M}(x). \tag{13}$$

We call the set of all cells which abide by this inequation $LAT(x)$. Finding a similar equation for longitudes is more demanding, as the equation depends on the latitude of cells c_i and c_j. Formally, the set $\tilde{M}(x)$ of y with $od(x, y) \leq \theta$ is clearly a subset of all y for which $|lat(x) - lat(y)| \leq \theta_R / \min\{cos(lon(x), lon(y))\}$ holds. Consequently, we can derive the following equation:

$$x \in c_i \wedge y \in c_j \wedge |c_i^{lon} - c_j^{lon}| > \left\lceil \frac{\alpha}{mincos(c_i, c_j)} \right\rceil \rightarrow y \notin \tilde{M}(x) \tag{14}$$

where

$$mincos(c_i, c_j) = \min\{cos(\alpha c_i), cos(\alpha(c_i + 1)), cos(\alpha(c_j)), cos(\alpha(c_j + 1)). \tag{15}$$

We call this set $LON(x)$. Now, if one the minimal cosine values in Equation 15 is 0, then Equation 14 is not well-defined. This happens when one of the cells c_i or c_j is adjacent to

one of the poles. In this case, we assume $\frac{\alpha}{mincos(c_i,c_j)} = 0$. This assumption has the simple consequence that we select all cells c_j at the poles to contain potential y with $od(x, y) \leq \theta$. We can now generate a first approximation of $\tilde{M}(x)$ by computing the intersection of all y that abide by Equations 13 and 14. We call this set $A(x) = LAT(x) \cap LON(x)$. Note that $\tilde{M}(x) \subseteq A(x)$. An example of such a set is shown in Figure 2 where A(Oslo) is depicted as a set of gray squares. Note that given that $\alpha = 1$, we only need to consider the cells with $28 \leq c_i^{lat} \leq 30$. Yet, given that cos(60?) = 0.5, the number of cells that have to be considered in longitude grows from 5 to 7 when crossing the 60th north parallel.

4.3 Optimality of Orchid for Points

While it is guaranteed that $\tilde{M}(x) \subseteq A(x)$, it is possible that $A(x)$ contains grid cell c with $\forall y \in c, (x, y, od(x, y)) \notin \tilde{M}$.[6] Such cells must be eliminated from $A(x)$ as they lead to unnecessary comparisons. Achieving this goal can be carried out by measuring the minimal distance from the cell $c(x)$ which contains x and all other cells $c \in A(x)$. Let us assume that c is at the north east of $c(x)$ (for reasons of symmetry, the argumentation can be extended to all other cells). In our example, such a cell would be that which contains Mora. Then the most north eastern point of $ne(c(x))$ has the coordinates $\Delta(c_i^{lat}(x) + 1, c_i^{lon}(x) + 1)$ while the most south western point of $sw(c)$ of c has the coordinates $\Delta(c_i^{lat}, c_i^{lon})$. Consequently, the minimal distance from points in $c(x)$ to points in c is

$$\min_{od}(c(x), c) = od(\Delta(c_i^{lat}(x) + 1, c_i^{lon}(x) + 1), \Delta(c_i^{lat}(x), c_i^{lon})). \tag{16}$$

We thus define the set $OPT(x) \subseteq A(x)$ as

$$OPT(x) = \{y \in A(x) : \min_{od}(c(x), c(y)) \leq \theta\}. \tag{17}$$

This set is guaranteed not to contain any cell with which elements of $c(x)$ should be compared. Consequently, it is the set of points x and all other elements of $c(x)$ are compared to by Orchid.

$OPT(x)$ is optimal in the sense that

$$\lim_{\alpha \to +\infty} OPT(x) = \tilde{M}(x). \tag{18}$$

This is simply due to $\alpha \to +\infty$ leading to $\Delta \to 0$. In this case, $c(x) = \{x\}$ and $c(y) = \{y\}$. Thus, $\min_{od}(c(x), c(y)) = od(x, y)$ which allows to infer that $OPT(x) = \tilde{M}(x)$ from Equation 17. Note that this proof shows that Orchid fulfills a necessary and sufficient condition to be reduction-ratio-optimal on single points in the sense of [10]. In our example $A(Oslo) = OPT(Oslo)$.

4.4 Comparing Polygons with Orchid

The extension of $OPT(x)$ to polygons is based on the following observation: Given the definition of Hausdorff distances,

$$hd(s, t) \leq \theta \rightarrow \forall s_i \in s \; \exists t_j \in t : od(s_i, t_j) \leq \theta \tag{19}$$

[6] Note that $od(x, y) = hd(x, y)$ for $|x| = |y| = 1$.

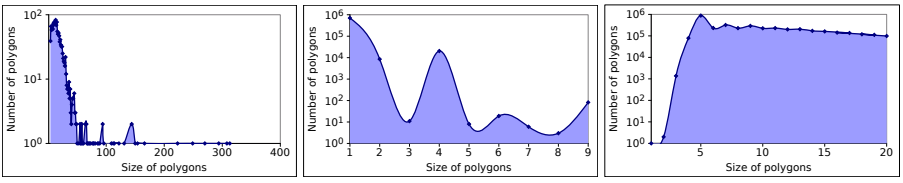
holds. Consequently, $OPT(s) = \bigcap_{s_i \in s} OPT(s_i)$. The reduction-ratio optimality of Orchid for polygons follows from its reduction-ratio-optimality for points.

5 Evaluation

The goal of the evaluation was to assess the performance of our approaches with respect to their runtime and the number of computation of the orthodromic distance that they carried out. To achieve this goal, we first compare the naive, bound, *CS* and *BC+CS* implementations of the computations of bound Hausdorff distance on samples from three different datasets. Note that we refrained from using the whole datasets because the runtime of the naive approach would have been impracticable. In the second part of our evaluation, we study the combination of Orchid and our Hausdorff implementations.

5.1 Experimental Setup

We selected three publicly available datasets of different sizes for our experiments. The first dataset, *Nuts*, contains a detailed description of 1,461 specific European regions.[7] The second dataset, *DBpedia*, contains all 731,922 entries from DBpedia that possess a geometry entry.[8] Finally, the third dataset, LGD, contains all 3,836,119 geo-spatial objects from LinkedGeoData that are instances of the class `Way`.[9] An overview of the distribution of the polygon sizes in these datasets is given in Figure 3. In addition, we used a dataset that consists of all points which have the `wgs84:geometry` property[10] from DBpedia for the comparison with SILK.[11] The 732,224 entities in this dataset are single points on the surface of the planet. We used this dataset because SILK 2.5.3 does not yet support the Hausdorff distance but implements the orthodromic distance.



(a) Nuts **(b)** DBpedia **(c)** LGD

Fig. 3. Distribution of polygon sizes

All experiments were carried out on a 32-core server running JDK 1.7 on Linux 10.04. The processors were 8 quadcore AMD Opteron 6128 clocked at 2.0 GHz. Unless stated otherwise, each experiment was assigned 10GB of memory and was ran 5

[7] We used version 0.9.1 as available at `http://nuts.geovocab.org/data/`

[8] We used version 3.8 as available at `http://dbpedia.org/Datasets`

[9] We used the RelevantWays dataset (version of April 26th, 2011) of LinkedGeoData as available at `http://linkedgeodata.org/Datasets`

[10] `wgs84` stands for `http://www.w3.org/2003/01/geo/wgs84_pos#`

[11] The dataset was extracted from the RelevantNodes dataset (version of April 26th, 2011) of DBpedia as available at `http://linkedgeodata.org/Datasets`

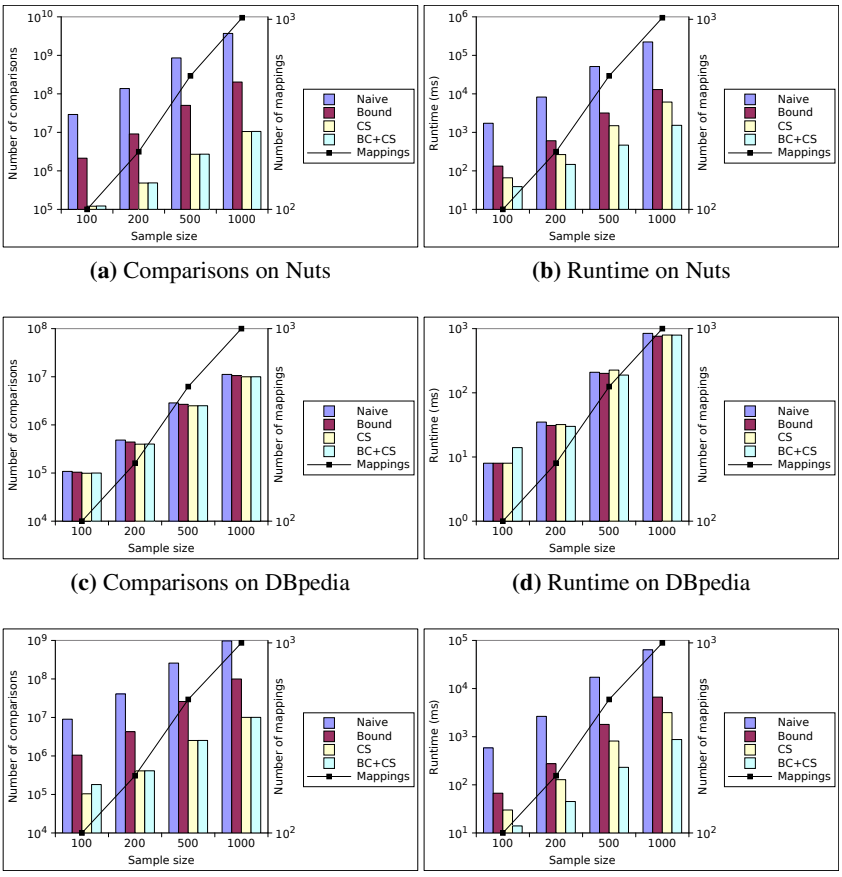
times. The time-out for experiments was set to 3 hours per iteration. The granularity parameter α was set to 1. In the following, we present the minimal runtime of each of the experiments.

5.2 Results

Hausdorff Implementations. In the first part of our evaluation, we measured the runtimes achieved by the three different implementation of the Hausdorff distances on random samples of the Nuts, DBpedia and LGD data sets. We used three different thresholds for our experiments, i.e., 100 m, 0.5 km and 1 km. In Figure 4, we present the results achieved with a threshold of 100 m. The results of the same experiments for 0.5 km and 1 km did not provide us with significantly different insights. All exact values can be found on the project website. As expected the runtime of all three approaches increases quadratically with the size of the sample. There is only a slight variation in the number of comparisons (see Figure 4) carried by the three approaches on the DBpedia dataset. This is simply due to most polygons in the dataset having only a small number of nodes as shown in Figure 3. With respect to runtime, there is no significant difference between the different approaches on DBpedia. This is an important result as it suggests that we can always use the CS or $BC + CS$ approaches even when the complexity of the polygons in the datasets is unknown.

On the two other datasets, the difference between the approaches with respect to both the number of comparisons and the runtime can be seen clearly. Here, the bound implementation requires an order of magnitude less comparisons than the naive approach while the indexed implementations need two orders of magnitude less comparisons. The runtimes achieved by the approaches reflect the observations achieved on the comparisons. In particular, the bound approach is an order of magnitude faster than the naive approach. Moreover, the $BC + CS$ approach outperforms the bound approach by approximately one further order of magnitude. Note that up to approximately 1.07% of the comparisons carried out by $BC + CS$ are the result of the indexing step.

Deduplication. In our second series of experiments, we deduplicated the three datasets at hand by using four different thresholds between 100 m and 2 km. We compared the combination of Orchid ($\alpha = 1$) and of all different implementations of the Hausdorff distance. The rationale behind this experiment was to measure whether the bound and indexed implementations were of any use even within the smaller sub-problems generated by Orchid. The results achieved show that using these implementations can indeed lead to significant improvements in both runtime and comparisons (see Figure 5). In particular, the indexed distance profits from the fact that it can discard a large number of computations that would lead to distance below and above the distance threshold. Thus, it requires over than two orders of magnitude less computations than the bound and naive versions on the Nuts dataset. Given the small size of the index that it generates for Nuts, the indexed approach is also two orders of magnitude faster across all the thresholds. On the LGD dataset, the indexed approach is the only one that terminated within the set time of 3 hours. Due to the topology of the DBpedia data, the runtimes on DBpedia are comparable for all approaches. Here, it is important to note that for smaller thresholds, the indexed approach still requires close to an order of magnitude less comparisons than the naive approach.



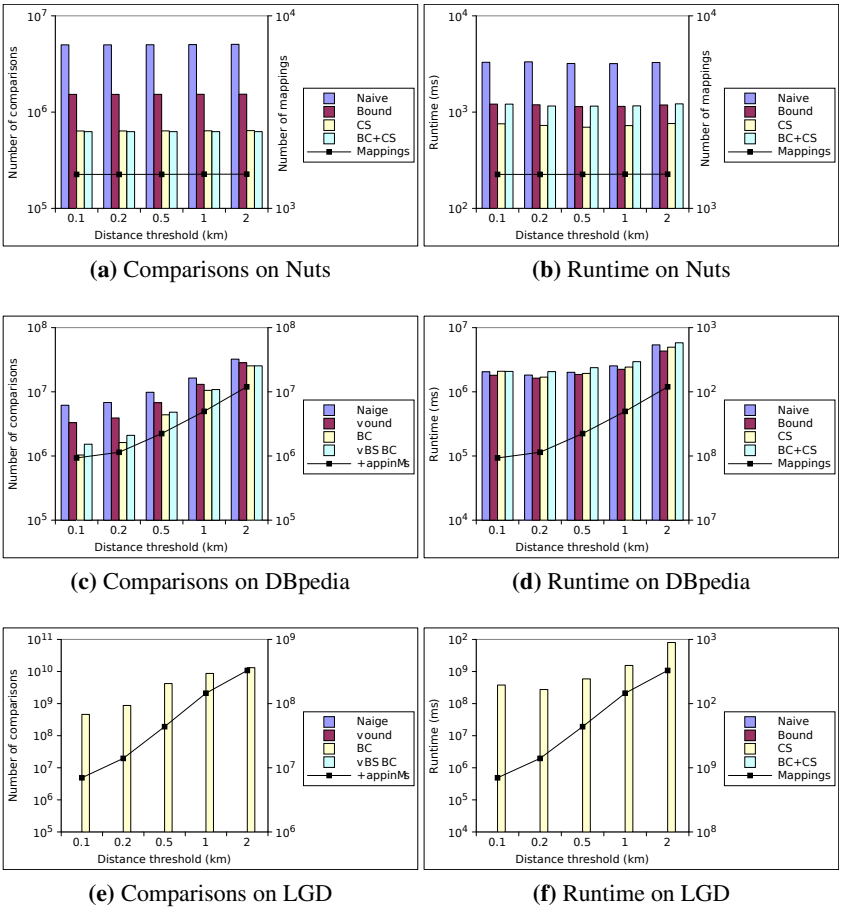
(a) Comparisons on Nuts **(b)** Runtime on Nuts

(c) Comparisons on DBpedia **(d)** Runtime on DBpedia

(e) Comparisons on LGD **(f)** Runtime on LGD

Fig. 4. Number of comparisons and runtimes on samples of the datasets

Scalability. We were also interested in knowing how our approach performs with growing dataset sizes. We thus ran Orchid in combination with *BC* with randomly selected slices of LinkedGeoData and DBpedia and computed the runtime against the size of the data slices. The similarity threshold was set to 0.1 *km* as in the previous experiment. The results on DBpedia and LinkedGeoData are shown in Table 1. We omitted Nuts because it is too small for scalability experiments. The runtimes and number of comparisons on DBpedia suggest that the approach behaves in a quasi-linear fashion on low-dimensional and sparsely distributed data. Note that the number of mappings because partly larger than the number of computations on this dataset is simply due to items with the same URI being found in both source and target and thus not necessitating any comparisons for deduplication. This is more rarely the case in the LinkedGeoData dataset. The runtimes on LinkedGeoData yet suggest that both the number of computations and the runtime required of our approach grow sub-linearly with the number of mappings to be computed when the number of points per polygon grows. This can be explained



(a) Comparisons on Nuts **(b)** Runtime on Nuts

(c) Comparisons on DBpedia **(d)** Runtime on DBpedia

(e) Comparisons on LGD **(f)** Runtime on LGD

Fig. 5. Number of comparisons and runtime of ORCHID

by our approach making effective use of existing data to discard computations and reduce the ratio of number of computations to mappings with growing data size. Thus, our approach promises to scale well to even larger data sets.

Comparison with Other Approaches. SILK[12] [6] is of the few other LD framework which implements the orthodromic distance. To the best of our knowledge, no other LD framework implements the Hausdorff distance. Thus, we compare ORCHID in combination with the naive implementation of the Hausdorff distance to SILK on all 732,224 points from DBpedia that contain longitude and latitude information. The results of four different distance thresholds are shown in Figure 6. Our results clearly show that ORCHID outperforms SILK by more than one order of magnitude in all settings.

[12] Throughout our experiments, we used SILK 2.5.3.

Table 1. Scalability results. The top section shows the results on DBpedia while the lower section shows the results on LinkedGeoData.

Sample Size	*od* computations	Runtime (ms)	Mappings
10^5	34,959	2,936	103,428
2×10^5	97,798	5,783	215,096
4×10^5	341,986	10,423	459,681
7.3×10^5	1,035,222	20,727	932,848
10^5	5,703,683	42,437	77,003
2×10^5	11,734,609	57,935	159,878
4×10^5	24,844,435	153,174	342,477
8×10^5	55,212,459	411,248	777,826
16×10^5	131,405,064	819,636	1,902,803

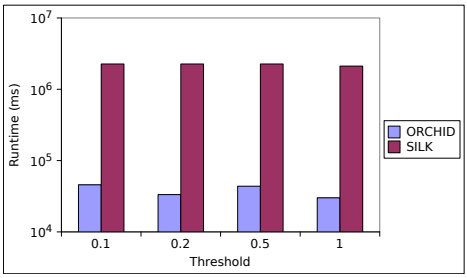


Fig. 6. Comparison of runtime of SILK and ORCHID

6 Related Work

The work presented herein is related to record linkage, deduplication, LD and the efficient computation of Hausdorff distances. An extensive amount of literature has been published by the database community on record linkage (see [7,4] for surveys). With regard to *time complexity*, time-efficient deduplication algorithms such as PPJoin+ [19], EDJoin [18], PassJoin [8] and TrieJoin [17] were developed over the last years. Several of these were then integrated into the hybrid LD framework LIMES [11]. Moreover, dedicated time-efficient approaches were developed for LD. For example, RDF-AI [15] implements a five-step approach that comprises the preprocessing, matching, fusion, interlink and post-processing of data sets. [12] presents an approach based on the Cauchy-Schwarz that allows discarding a large number of unnecessary computations. The approaches HYPPO [9] and $\mathcal{HR}^3$ [10] rely on space tiling in spaces with measures that can be split into independent measures across the dimensions of the problem at hand. Especially, $\mathcal{HR}^3$ was shown to be the first approach that can achieve a relative reduction ratio r' less or equal to any given relative reduction ratio $r > 1$. Standard blocking approaches were implemented in the first versions of SILK and later replaced with MultiBlock [6], a lossless multi-dimensional blocking technique. KnoFuss [13] also implements blocking techniques to achieve acceptable runtimes.

Hausdorff distances are commonly used in fields such as object modeling, computer vision and object tracking. [1] presents an approach for the efficient computation of Hausdorff distances between convex polygons. While the approach is quasi-linear in the number of nodes of the polygons, it cannot deal with non-convex polygons as commonly found in geographic data. [5] presents an approach for the comparison of 3D models represented as triangular meshes. The approach is based on a subdivision sampling algorithm that makes use of octrees to approximate the distance between objects. [16] present a similar approach that allows approximating Hausdorff distances within a certain error bound while [3] presents an exact approach. [14] present an approach to compute Hausdorff distances between trajectories using R-trees within an L_2-space. Note that our approach is tailored to run in orthodromic spaces. Still, some of the insights presented in [14] may be usable in an orthodromic space. To the best of our knowledge, none of the approaches proposed before address the problem of finding pairs of polygons (A, B) such that $hd(A, B) \leq \theta$ in an orthodromic space.

7 Conclusion and Future Work

In this paper, we presented Orchid, a LD approach for geographic data. Our approach is based on the combination of Hausdorff and orthodromic distances. We devised two approaches for computing bound Hausdorff distances and compared these approaches with the naive approach. Our experiments showed that we can be more than two orders of magnitude faster on typical geographic datasets such as Nuts and LinkedGeoData. We then presented the space tiling approach which underlies Orchid and proved that it is reduction-ratio-optimal. Our most interesting result was that our approach seems to be sub-linear with respect to the number of comparisons and the runtime it requires. This behavior can be explained by the approach making use of the higher data density to perform better distance approximations and thus discarding more computations of the orthodromic distance. In addition to comparing different parameter settings of Orchid with each other, we also compared our approach with the state-of-the-art LD framework SILK. Our results show that we outperform the blocking approach it implements by more than one order of magnitude. In future work, we will extend our approach by implementing it in parallel and integrating it with a load balancing approach.

Acknowledgement. The work presented in this paper was financed by the EU-FP7 Project GeoKnow (Grant Agreement No. 318159).

References

1. Atallah, M.J.: A linear time algorithm for the hausdorff distance between convex polygons. Technical report, Purdue University, Department of Computer Science (1983)
2. Atallah, M.J., Ribeiro, C.C., Lifschitz, S.: Computing some distance functions between polygons. Pattern Recognition 24(8), 775–781 (1991)
3. Bartoň, M., Hanniel, I., Elber, G., Kim, M.-S.: Precise hausdorff distance computation between polygonal meshes. Comput. Aided Geom. Des. 27(8), 580–591 (2010)

4. Elmagarmid, A.K., Ipeirotis, P.G., Verykios, V.S.: Duplicate record detection: A survey. IEEE Trans. Knowl. Data Eng. 19(1), 1–16 (2007)
5. Guthe, M., Borodin, P., Klein, R.: Fast and accurate hausdorff distance calculation between meshes. J. of WSCG 13, 41–48 (2005)
6. Isele, R., Jentzsch, A., Bizer, C.: Efficient Multidimensional Blocking for Link Discovery without losing Recall. In: WebDB (2011)
7. Köpcke, H., Rahm, E.: Frameworks for entity matching: A comparison. Data Knowl. Eng. 69(2), 197–210 (2010)
8. Li, G., Deng, D., Wang, J., Feng, J.: Pass-join: a partition-based method for similarity joins. Proc. VLDB Endow. 5(3), 253–264 (2011)
9. Ngonga Ngomo, A.C.: A Time-Efficient Hybrid Approach to Link Discovery. In: OM 2011 (2011)
10. Ngonga Ngomo, A.-C.: Link discovery with guaranteed reduction ratio in affine spaces with minkowski measures. In: Cudré-Mauroux, P., Heflin, J., Sirin, E., Tudorache, T., Euzenat, J., Hauswirth, M., Parreira, J.X., Hendler, J., Schreiber, G., Bernstein, A., Blomqvist, E. (eds.) ISWC 2012, Part I. LNCS, vol. 7649, pp. 378–393. Springer, Heidelberg (2012)
11. Ngonga Ngomo, A.-C.: On link discovery using a hybrid approach. J. Data Semantics 1(4), 203–217 (2012)
12. Ngonga Ngomo, A.-C., Auer, S.: LIMES - A Time-Efficient Approach for Large-Scale Link Discovery on the Web of Data. In: IJCAI, pp. 2312–2317 (2011)
13. Nikolov, A., d'Aquin, M., Motta, E.: Unsupervised learning of link discovery configuration. In: Simperl, E., Cimiano, P., Polleres, A., Corcho, O., Presutti, V. (eds.) ESWC 2012. LNCS, vol. 7295, pp. 119–133. Springer, Heidelberg (2012)
14. Nutanong, S., Jacox, E.H., Samet, H.: An incremental hausdorff distance calculation algorithm. Proc. VLDB Endow. 4(8), 506–517 (2011)
15. Scharffe, F., Liu, Y., Zhou, C.: Rdf-ai: an architecture for rdf datasets matching, fusion and interlink. In: Proc. IJCAI 2009 Workshop on Identity, Reference, and Knowledge Representation (IR-KR), Pasadena, CA, US (2009)
16. Tang, M., Lee, M., Kim, Y.J.: Interactive hausdorff distance computation for general polygonal models. ACM Trans. Graph. 28(3), 74:1–74:9 (2009)
17. Wang, J., Li, G., Feng, J.: Trie-join: Efficient trie-based string similarity joins with edit-distance constraints. PVLDB 3(1), 1219–1230 (2010)
18. Xiao, C., Wang, W., Lin, X.: Ed-Join: an efficient algorithm for similarity joins with edit distance constraints. PVLDB 1(1), 933–944 (2008)
19. Xiao, C., Wang, W., Lin, X., Yu, J.X.: Efficient similarity joins for near duplicate detection. In: WWW, pp. 131–140 (2008)