RADON — Rapid Discovery of Topological Relations

Mohamed Ahmed Sherif, Kevin Dreßler, Panayiotis Smeros, Axel-Cyrille Ngonga Ngomo

Department of Computer Science, University of Leipzig, 04109 Leipzig, Germany

sherif@informatik.uni-leipzig.de

dressler@informatik.uni-leipzig.de

Panayiotis Smeros, Axel-Cyrille Ngonga Ngomo

EPFL, BC 142, Station 14, CH-1015 Lausanne, Switzerland

panayiotis.smeros@epfl.ch

Abstract

Geospatial data is at the core of the Semantic Web, of which the largest knowledge base contains more than 30 billions facts. Reasoning on these large amounts of geospatial data requires efficient methods for the computation of links between the resources contained in these knowledge bases. In this paper, we present RAox – efficient solution for the discovery of topological relations between geospatial resources according to the DE-9IM standard. Our evaluation shows that we outperform the state of the art significantly and by several orders of magnitude.

1 Introduction

Geo-spatial datasets belong to the largest sources of Linked Data. For example, LinkedGeoData\(^1\) contains more than 20 billion triples which describe millions of geo-spatial entities. Datasets such as NUTS\(^2\) use polygons of up to 1500 points to describe resources such as countries. As pointed out in previous works (Ngonga Ngomo 2013), only 7.1\% of the links between resources connect geo-spatial entities. This is due to two main factors. First, the large number of geo-spatial resources available on the Linked Data Web requires scalable algorithms for computing links between geo-spatial resources. In addition, the description of geo-spatial resources being commonly based on polygons demands the computation of particular relations, i.e., topological relations, between geo-spatial resources. According to the Linked Data principles\(^3\) and for the sake of real-time application such as structured machine learning (e.g., DL-Learner (Lehmann 2009)) and question Answering (e.g., DEQA platform (Lehmann et al. 2012)), the provision of explicit topological relations between resources is of central importance to achieve scalability. However, only a few approaches have been developed to deal with geo-spatial data represented in RDF. For example, (Ngonga Ngomo 2013) uses the Hausdorff distance to compute a topological distance between geo-spatial entities. (Smeros and Koubarakis 2016) builds upon MultiBlock to compute topological relations according to the DE-9IM standard.

We go beyond the state of the art by providing a novel indexing method combined with space tiling that allows for the efficient computation of topological relations between geo-spatial resources. In particular, we present a novel sparse index for geo-spatial resources. We then develop a strategy to discard unnecessary computations for DE-9IM relations based on bounding boxes. Our extensive experiments show that our approach scales well and outperforms the state of the art by up to 3 orders of magnitude w.r.t. to its runtime. The contributions of this paper can be summarized as follows: (1) We present a novel indexing algorithm for geo-spatial resources based on an optimized sparse space tiling. (2) We provide a novel filtering approach for the rapid discovery of topological relations (RADON), which uses minimum bounding box (MBB) approximation. (3) We show that RADON is able to discover any of the DE-9IM relations that involve intersection of at least one point. (4) We evaluate RADON on real datasets and show that it clearly outperforms the state of the art.

2 Preliminaries

Let \(K\) be a finite RDF knowledge base. \(K\) can be regarded as a set of triples \((s, p, o)\) \(\in (R U B) \times P \times (R U L U B)\), where \(R\) is the set of all resources, \(B\) is the set of all blank nodes, \(P\) the set of all predicates and \(L\) the set of all literals. Given a set of source resources \(S\) and target resources \(T\) from two (not necessarily distinct) knowledge bases \(K_1\) and \(K_2\) as well as a relation \(R\), the goal of Link Discovery (LD) is to find the set of mapping \(M = \{(s, t) \in S \times T : R(s, t)\}\). Naive computation of \(M\) requires quadratic time complexity to compare every \(s \in S\) with every \(t \in T\), which is clearly impracticable for large datasets such as geo-spatial datasets, which are the focus of this work. Here, we present an algorithm for efficient computations of topological relations between resources with geo-spatial descriptions (i.e., described by means of vector geometry).\(^4\) We assume that each of the resources in \(S\) and \(T\) considered in the subsequent portion of this paper as being described by a geometry, where each geometry is modelled as sequence of points. An example of such resources is shown in Figure 1(a).

The Dimensionally Extended nine-Intersection Model

Copyright © 2017, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.

\(^1\)http://linkedgeodata.org

\(^2\)http://nuts.geovocab.org/

\(^3\)https://www.w3.org/DesignIssues/LinkedData.html

\(^4\)Most commonly encoded in the WKT format, see http://www.opengeospatial.org/standards/sfa.
(DE-9IM) (Clementini, Sharma, and Egenhofer 1994) is a standard used to describe the topological relations between two geometries in two-dimensional space. The spatial relations expressed by the model are topological and are invariant to rotation, translation and scaling transformations (Egenhofer and Franzosa 1991). The basic idea behind DE-9IM is to construct the $3 \times 3$ intersection matrix:

$$
\begin{bmatrix}
\text{dim}(g_1 \cap I) & \text{dim}(g_1 \cap B) & \text{dim}(g_1 \cap E) \\
\text{dim}(B \cap g_2) & \text{dim}(B \cap g_2) & \text{dim}(B \cap g_2) \\
\text{dim}(g_1 \cap E) & \text{dim}(g_1 \cap E) & \text{dim}(g_1 \cap E)
\end{bmatrix}
$$

(1)

where $\text{dim}$ is the maximum number of dimensions of the intersection $\cap$ of the interior $(I)$, boundary $(B)$, or exterior $(E)$ of the two geometries $g_1$ and $g_2$. The domain of $\text{dim}$ is $\{-1, 0, 1, 2\}$, where $-1$ indicates no intersection, $0$ stands for an intersection which results into a set of one or more points, $1$ indicates an intersection made up of lines and $2$ standard for an intersection which results in an area. A simplified binary version of $\text{dim}(x)$ with the binary domain $\{true, false\}$ is obtained using the boolean function $\beta(\text{dim}(g)) = false$ iff $\text{dim}(g) = -1$ and true otherwise.

The major insight behind RADO is that one condition must hold for any of the entries of the DE-9IM matrix to be true: There must be at least one point in space that is common to the perimeters of the polygons. Note that the only spatial relation for which all entries are $0$ is the disjoint relation, which RADO can easily compute by computing the inverse of the intersects relation. Hence, by accelerating the computation of whether two geometries share at least one point, we can accelerate the computation of any of the DE-9IM entries. Therewith, we can also accelerate the computation of any topological relation, as they can all be derived from the DE-9IM entries. We implement this insight by using an improved indexing approach based on minimum bounding boxes and space tiling.

The minimum bounding box (MBB) of a geometry $g$ in $n$ dimensions (O’Rourke 1985) (also called its envelope) is the rectangular box with the smallest measure (area, volume, or hypervolume in higher dimensions) within which all points of $g$ lie. Let $k_i(p)$ denote the $i$th dimension coordinate of a point $p$. To obtain the MBB of a geometry $g$, we have to find the lowest point coordinate $c_1 = \min_{p \in g} k_1(p)$ and the highest point coordinate $c_T = \max_{p \in g} k_1(p)$ in each dimension $i \in \{0, \ldots, n\}$. Then, the $2^n$ vertices of the MBB in $n$ dimensions are all the vectors $(c_0^{(i)}, c_1^{(i)}, \ldots, c_T^{(i)})$, where $(\cdot)^{(i)} \in \{\perp, \top\}$.

Figure 1(b) shows an example of using the MBB to abstract the running example in Figure 1(a).

Space tiling is an indexing technique for spatial data inspired by tessellation and previously used by LD optimization approaches such as ORU (Ngonga Ngomo 2013) and HR$^2$ (Ngonga Ngomo 2012). The main idea behind space tiling is to divide $n$-dimensional affine spaces into arbitrarily many hypercubes with the same edge length $\ell$. These hypercubes are indexed with vectors $i \in \mathbb{N}$ to serve as addressable buckets for geometries. In turn, the obtained index structures can be exploited by various optimization techniques. We call $\Delta = \ell^{-1}$ the granularity factor. This notion of space tiling can be generalized to hyperrectangles, in which case there exist $n$ independent granularity factors $\Delta_i$ where $i \in \{0 \ldots n\}$.

Note that although we eventually use hyperrectangles, we will stick to the term hypercube for the sake of simplicity and just define independent granularity factors when necessary. Figure 1(c) shows our running example along with a grid of hypercubes using $\Delta = 2$, where the green area will be indexed to each highlighted hypercube.

## 3 Approach

We have now introduced all ingredients necessary for defining the RADO algorithm (Algorithm 1). RADO takes a set of source resources $S$, a set of target resources $T$ and a topological relation $r$ as input. The goal of RADO is to generate the mapping $M = \{(s, t) \in S \times T : r(s, t)\}$ efficiently, where $r$ is a topological relation. RADO addresses this challenge by means of three optimization steps: Swapping for index size minimization, Space tiling for indexing and filtering to improve the runtime of the computation of topological relations. In the following, we detail each of these steps.

### 3.1 Swapping Strategy

We introduce the Estimated Total Hypervolume (ETH) of a set of geometries $X$ as

$$
\text{ETH}(X) = |X| \prod_{i=1}^{d} \frac{1}{|X_i|} \sum_{p \in X} \left( \max_{p \in X} k_i(p) - \min_{p \in X} k_i(p) \right),
$$

(2)

with $d$ being the number of dimensions of the resource geometries and $k_i(p)$ denoting the coordinate of a point $p$ in the $i$th dimension. If $\text{ETH}(T) < \text{ETH}(S)$, RADO swaps $S$ and $T$ and computes the reverse relation $r'$ instead of $r$ (Lines 2–5). For example, if $r$ were the topological relation covered and $\text{ETH}(S) < \text{ETH}(T)$, then RADO swaps $S$ and $T$ and compute the reverse relation of $r$, i.e., coveredBy. The rationale behind using ETH instead of the size of the datasets is that even small datasets can contain very large geometries that span over a large number of hypercubes and would lead to large

---

$^3$Formally, the reverse relation $r'$ of a relation $r$ is defined as $r'(y, x) \Leftrightarrow r(x, y)$. 

---

176
spatial index when used as source. For the sake of illustration, consider the running example in Figure 1(a). Here, we can see that the \texttt{eth} of NUTS (containing only the gray geometry) is greater than the \texttt{eth} of CLC (containing the green and blue areas). Thus, we set \( S = \text{CLC} \) and \( T = \text{NUTS} \).

### 3.2 Optimized Sparse Space Tiling

In its second step, \textsc{Radon} utilizes space tiling to insert all geometries \( s \in S \) and \( t \in T \) into an index \( I \), which maps resources to sets of hypercubes. Let \( \Delta_x \) and \( \Delta_y \) be the granularities across the latitude and longitude (several strategies can be used to compute these values. We present and evaluate them in Section 4). For indexing a resource \( x \), we begin by computing its MMB’s upper left and lower right corners coordinates \((\phi_1(x), \lambda_1(x))\) and \((\phi_2(x), \lambda_2(x))\) respectively (Line 8). Then, we map each \( x \) to all hypercubes over which its MBB spans (Lines 9–11). To this end, we transform the MBB’s corner coordinates into hypercube indices using \( \psi_x \) and \( \psi_y \) from Equation 3.

\[
\psi_x^i(x) = [x \cdot \Delta_x^i] \quad \psi_y^j(x) = [x \cdot \Delta_y^j]
\]

We then map \( x \) to all hypercubes with indices \((i, j)\) where \( i, j \in \mathbb{Z} \), \( \psi_x^i(\phi_1(x)) \leq i \leq \psi_x^i(\phi_2(x)) \) and \( \psi_y^j(\lambda_1(x)) \leq j \leq \psi_y^j(\lambda_2(x)) \). Note that the special case of geometries passing over the antimeridian is detected and dealt with by splitting such geometries into 2 geometries before and after the antimeridian. The index \( I \) now contains the portions of the space (i.e., the hypercubes) within which portions of \( x \) can potentially be found. It is important to notice that entities in portions of space that do not belong to the hypercubes which contain elements of \( S \) (denoted \( I(S) \)) will always be disjoint with the elements of \( T \). We leverage this insight as follows: We first index all \( s \in S \). Then we follow the same procedure for \( t \in T \) (Lines 14–21) but only index geometries \( t \) that are potentially in hypercubes already contained in \( I(S) \). This optimized sparse space tiling is the motivation for the previously introduced swapping strategy. Indexing the dataset with the least \texttt{eth} first results in an index \( I \) with less hypercubes.

Consider again our running example in Figure 1(c) for the sake of illustration. Assume the granularity factors are \( \Delta_x = \Delta_y = 2 \). The green area’s MBB has the following corner coordinates: \((\phi_1(g), \lambda_1(g)) = (12.340703846780286, 51.28797110806819)\) and \((\phi_2(g), \lambda_2(g)) = (12.389192648396918, 51.33902633403139)\). Therefore, \( \psi_x^i(\phi_1(g)) = 24, \psi_x^i(\phi_2(g)) = 25, \psi_y^j(\lambda_1(g)) = 102, \psi_y^j(\lambda_2(g)) = 103 \) and thus this geometry will be indexed into the four highlighted hypercubes with index vectors \((24, 102), (24, 103), (25, 102) \) and \((25, 103)\). In Figure 1(d), we highlighted all hypercubes containing the gray geometry after the optimized sparse space tiling. Notice that many hypercubes are empty as a result of not containing any portion of the other dataset’s geometries.

### 3.3 Link Generation

After the computation of the index \( I \), \textsc{Radon} implements the last speedup strategy using a MBB-based filtering technique. For each hypercube with indexed geometries from both \( S \) and \( T \) (Line 24), \textsc{Radon} first discards unnecessary computations using the TestMMB procedure. TestMMB optimizes the subset of DE-9IM relations for relations where one geometry has interior or boundary points in the exterior of the other geometry, i.e., \( s \subseteq t \) or \( t \subseteq s \) (e.g., \texttt{equals}, \texttt{covers}, within formally defined in the annex (Sherif et al. 2016)). Let \( \mathcal{O}(g) \) denote the MBB geometry of a geometry \( g \). Note that \( g \subseteq \mathcal{O}(g) \) always holds. We can now infer \( \neg r(\mathcal{O}(s), \mathcal{O}(t)) \rightarrow \neg r(s, t) \) using the transitivity of \( \subseteq \). For all other relations, TestMMB simply returns \texttt{true}. For example, in our running example in Figure 1(b), if \( r \) is the within topological relation, we do not need to compute \( r \) for the blue geometry, as its MBB is not completely within the gray geometry’s MBB. In case the TestMMB method returns \texttt{true}, \textsc{Radon} carries out the more expensive computation of the topological relation between the geometries \( s \) and \( t \) (Line 31). If \( r(s, t) \) holds, \textsc{Radon} adds the pair \((s, t)\) to the result mapping \( M \). To make sure that we compute each pair \((s, t) \in S \times T \) at most once, we cache the computed pairs \((s, t)\) in the mapping \( C \) (Lines 27–29).

**Proposition 1.** \textsc{Radon} is complete and correct w.r.t. the application of space tiling.

### 4 Evaluation

**Topological relations** Only a subset of the topological relations obtainable through DE-9IM reflects the semantics of the English language (Clementini, Di Felice, and van Oosterom 1993; Clementini, Sharma, and Egenhofer 1994) including \texttt{equals}, \texttt{within}, \texttt{contains}, \texttt{disjoint}, \texttt{touches}, \texttt{meets}, \texttt{covers}, \texttt{coveredBy}, \texttt{intersects}, \texttt{inside}, \texttt{crosses} and \texttt{overlaps}. Note that some of these relations are \texttt{synonyms} (e.g., \texttt{touches}(x, y) \iff \texttt{meets}(x, y)) while others are \texttt{combinations} of more atomic relations (e.g., \texttt{equals}(x, y) \iff \texttt{within}(x, y) \land \texttt{contains}(x, y)). Moreover, some relations are the \texttt{reverse} of some other relation. Hence, in this evaluation, we focused on the rapid computation of those 7 relations.\(^7\) In our running example in Figure 1(a), if we dub the blue, green and gray areas in \( a_1, a_2 \) and \( a_3 \) respectively. Then, \( \texttt{distinct}(a_1, a_2), \texttt{within}(a_2, a_3) \) and \( \texttt{intersects}(a_2, a_3) \) hold.

**Hardware and Software** All experiments were carried out on a 64-core 2.3 GHz PC running OpenJDK 64-Bit Server 1.7.0 75 on \textit{Ubuntu} 14.04.2 LTS. Each experiment was assigned 20 GB RAM and a timeout limit of 2 hour. Experiments which ran longer than this upper limit were terminated and the processed data percentage as well as the estimated time are reported. We evaluate \textsc{Radon} against two state-of-the-art approaches: (1) \textsc{Silk} as it is (to the best of our knowledge) the only LD framework that supports the discovery of topological relation, (2) \textsc{Strabon} as it implements the standard GeoSPARQL (OGC 2010) and is based on PostGIS. For \textsc{Silk} experiments, we ran our experiments using its latest version (v2.6.1) with a blocking factor of 10 as in (Smers and Koubarakis 2016). For \textsc{Strabon}, we also used the latest version (v3.2.10) with the accordingly tuned

---

\(^6\)Proof is given in the annex (Sherif et al. 2016).

\(^7\)See the annex (Sherif et al. 2016) for the formal definitions.
Algorithm 1: Radon

\textbf{input} : \( S \), set of source resources. \( T \), set of target 
resources. \( r \), topological relation. 
\textbf{output} : \( M \), Mapping \( \{(s, t) \in S \times T : r(s, t)\} \)

1. reversed \leftarrow false;
2. \textbf{if} \ethfi(T) < \ethfi(S) \textbf{then}
3. \hspace{1em} swap(S, T);
4. \hspace{1em} \( r \leftarrow r' \);
5. \hspace{1em} reversed \leftarrow true;
6. \( (\Delta_\alpha, \Delta_\lambda) \leftarrow \text{FindBestGranularity}(S, T) \);
7. \hspace{1em} \textbf{foreach} geometry \( s \in S \) \textbf{do}
8. \hspace{2em} (\phi_1(s), \lambda_1(s), \phi_2(s), \lambda_2(s)) \leftarrow 
\hspace{2em} \text{GetMBBDiagonalCorners}(s);
9. \hspace{2em} \textbf{for} \( i \leftarrow [\phi_1(s) \cdot \Delta_\phi] \) \textbf{to} \([\phi_2(s) \cdot \Delta_\phi] \) \textbf{do}
10. \hspace{3em} \textbf{for} \( j \leftarrow [\lambda_1(s) \cdot \Delta_\lambda] \) \textbf{to} \([\lambda_2(s) \cdot \Delta_\lambda] \) \textbf{do}
11. \hspace{4em} \text{InsertIntoHypercube}(I(S), i, j, s);
12. \hspace{4em} \( i \leftarrow i + 1; \)
13. \hspace{1em} \( i \leftarrow i + 1; \)
14. \hspace{1em} \textbf{foreach} geometry \( t \in T \) \textbf{do}
15. \hspace{2em} (\phi_1(t), \lambda_1(t), \phi_2(t), \lambda_2(t)) \leftarrow 
\hspace{2em} \text{GetMBBDiagonalCorners}(t);
16. \hspace{2em} \textbf{for} \( i \leftarrow [\phi_1(t) \cdot \Delta_\phi] \) \textbf{to} \([\phi_2(t) \cdot \Delta_\phi] \) \textbf{do}
17. \hspace{3em} \textbf{for} \( j \leftarrow [\lambda_1(t) \cdot \Delta_\lambda] \) \textbf{to} \([\lambda_2(t) \cdot \Delta_\lambda] \) \textbf{do}
18. \hspace{4em} \textbf{if} \text{GetHypercube}(I(S), i, j) \text{ is not empty} \textbf{then}
19. \hspace{5em} \text{InsertIntoHypercube}(I(T), i, j, t);
20. \hspace{5em} \( j \leftarrow j + 1; \)
21. \hspace{5em} \( i \leftarrow i + 1; \)
22. \hspace{1em} \textbf{foreach} geometry \( h_T \in I(S) \) \textbf{do}
23. \hspace{2em} \( H_T \leftarrow \text{GetHypercube}(I(s, t), \varphi(H_T), \lambda(H_T)) \);
24. \hspace{2em} \textbf{if} \( H_T \) \text{ is not empty} \textbf{then}
25. \hspace{3em} \textbf{for} \( s \in H_T \) \textbf{do}
26. \hspace{4em} \textbf{if} \( (s, t) \notin C \textbf{ then} \)
27. \hspace{5em} \( C \leftarrow C \cup \{s, t\} \);
28. \hspace{5em} \( B \leftarrow (\phi_1(s), \lambda_1(s), \phi_2(s), \lambda_2(s)) \), 
\hspace{5em} \( (\phi_1(t), \lambda_1(t), \phi_2(t), \lambda_2(t)) \); 
29. \hspace{5em} \textbf{if} TestMBB(r, B) \textbf{ then}
30. \hspace{6em} \textbf{if} r(s, t) \text{ is true then}
31. \hspace{7em} \text{M} \leftarrow M \cup \{s, t\};
32. \hspace{1em} \textbf{else} \textbf{ return } M' ;
33. \hspace{1em} \textbf{else} \textbf{ return } M ;

PostgreSQL (v9.1.13) and PostGIS (v2.0) as proposed by the 
developers. A more complete list of results can be obtained 
from the project website\(^8\). Note that Radon achieves a precision, 
a recall and an F-measure of 1 by virtue of its complete-

\(^8\)http://aksw.org/Projects/LIMES

ness and correctness. SILK and STRABON theoretically achieve 
the same F-measure (we were not always able to check this 
value for the two systems as the experiments did not always 
terminate before the timeout).

**Datasets** We evaluated our approach using two real-world 
datasets. (1) NUTS\(^9\) is manually curated by the Eurostat 
group of the European Commission. NUTS contains a 
detailed hierarchical description of statistical regions for the 
whole European regions. (2) CORINE Land Cover or simply 
CLC is an activity of the European Environment Agency 
that collects data regarding the land cover of European 
countries. CLC contains 44 sub-datasets range in size from 240 
to 248,242 resources.\(^10\) For testing the scalability of Radon, 
we merged all subsets of CLC into one big dataset of size 
2,209,538 (dubbed CLC\(_m\)). We preprocessed the datasets in 
the following fashion: To enable the processing of the NUTS 
dataset by Radon, SILK and STRABON, the ngeo:posList 
serialization was converted into the WKT format prior to 
experiments. Moreover, because of a SILK issue\(^11\), we had to 
trim lines larger than 64 KB from all datasets in order to get 
a fair comparison. All the reported dataset sizes are after 
preprocessing.

**Granularity Factor Selection Heuristic** The aim of this 
experiment was to evaluate different heuristics to approximate 
the optimal granularity factors \( \Delta_\phi \) and \( \Delta_\lambda \) used 
for tiling the space and generating the sparse index of hyper-
cubes. We tried 4 different heuristics corresponding to a 
statistical measure: \textit{minimum}, \textit{maximum}, \textit{median} and \textit{average}. 
Each heuristic first computes the respective statistical measure \( \eta \) independently for both datasets and both dimensions, 
resulting in 4 temporary values \( h_{\eta,\phi}(S), h_{\eta,\phi}(T), h_{\eta,\lambda}(S), h_{\eta,\lambda}(T) \). Finally, the granularity factor in each dimension 
is the average of the two datasets. Formally,

\[ h_{\eta,\phi}(X) = \eta \left\{ \frac{\max(\varphi(p)) - \min(\varphi(p))}{p \in X} \right\} \] \hspace{1em} (4)

\[ \Delta_{\eta,\phi}(S, T) = \frac{1}{2} \left( h_{\eta,\phi}(S) + h_{\eta,\phi}(T) \right) \] \hspace{1em} (5)

\( h_{\eta,\lambda}(X) \) and \( \Delta_{\eta,\lambda}(S, T) \) are defined similarly. \( S, T \) are 
the input source and target datasets, \( \varphi(p) \) the latitude of a point \( p \), \( \lambda(p) \) the longitude of \( p \) and \( \eta \in \{\min, \max, \text{median}\} \). 
We used all the 44 subsets of the CLC dataset as input for 
this experiment and recorded how many times each heuris-
tic achieved the best runtime for the \textit{intersects} relation. 
Additionally, when a heuristic was not the best in a run, we 
computed the percentage it was worse than the best one. The 
\textit{average} heuristic achieves the best result 24 times out of 44 
experiments. Runner-up is \textit{median}, achieving the best run-
time 17 times. Finally, the \textit{min} and \textit{max} heuristics achieved 
only 2 and 1 time(s) respectively. Interestingly, \textit{average} and 
\textit{median} were only 4% slower than the best measure on aver-
age when not being the best, while \textit{min} and \textit{max} where 34%
and 61% worse on average respectively. Based on these results, we used the average heuristic as the granularity selection policy in the rest of the experiments.

The basic idea behind the first three sets of experiments is to quantify the speedup gained by Radon over other LD frameworks. To the best of our knowledge, only the SilK LD framework recently (Smeros and Koubarakis 2016) implemented a multi-dimensional blocking approach to compute the topological relations. Therefore, we compare Radon’s and SilK’s runtimes in the subsequent experiments.

In the first set of experiments, we aimed to quantify the speedup of Radon over the other state-of-the-art approaches when applied to small datasets. To this end, we ran 44 experiments for each of the 7 basic topological relations identified in the previous section. In each experiment, we compared one of the 44 subsets of the CLC with the full NUTS. Altogether, we carried out 308 experiments. Note that both Radon and SilK were run on 1 core. Radon achieves an average speedup of 221.52, 213.76, 4.94, 4.82, 4.77, 4.76 and 4.75 times faster than SilK for the relations within, equals, covers, overlaps, intersects, crosses and touches respectively. Overall, Radon was able to outperform SilK by being 65.62 times faster on average over all topological relations. Moreover, Radon was able to achieve a linear speedup relative to the dataset sizes. In Figure 2, we show an overview of a subset of the experimental results (including a linear fit) achieved on the relations on which Radon achieved the best (up to two orders of 450 times faster) and the poorest (up to 6.5 times faster) relative performance w.r.t. SilK. Figures for other relations are given in the annex (Sherif et al. 2016). Moreover, Radon ran significantly less complete computations of the relations at hand. For example in Figure 3(a), Radon carries out only 3 and 4 computations for the “equals” and “within” relations respectively. On average, 449 times less computations per relation.

In the second set of experiments we aimed to evaluate the scalability of Radon when applied to big datasets. Thus, we used the merged dataset CLC\textsubscript{m} as both source and target dataset and ran Radon and SilK on 1 core. For more complete results, see Table 1 in the annex (Sherif et al. 2016). Radon is able to finish all the tasks within 67.44 minutes on average (maximum = 95.10 minutes for the crosses relation). SilK was only able to (on average) finalize 0.34% of each task within the 2 hour timeout limit. We extrapolated the runtime of SilK linearly to get an approximation of how long it would need to carry out the tasks at hand. On average, SilK would need 24.85 days to complete each task (linear extrapolation). Consequently, Radon is at least 715 times faster than SilK on average. These results emphasize the ability of Radon to deal with large datasets even when ran on 1 core.

In the third set of experiments, we wanted to quantify the speedup gained by using a parallel implementation of Algorithm 1 over the parallel implementation of SilK. For load balancing in Radon, we used the simple round robin load balancing policy (Shreedhar and Varghese 1996) with chunks size of 1000. As data, we used CLC\textsubscript{m} as both source and target. The parallel implementations were configured to run using 2, 4 and 8 threads. The results (For detailed result see Table 1 in the annex (Sherif et al. 2016).) show that our parallel implementation for Radon was able to discover all the topological relations in 20.83 minutes on average (maximum of 49.03 minutes in the case of the intersects relation). On the other side, SilK implementation was only able to (on average) finalize 1.16% of each task within the 2 hours timeout limit. We extrapolated the performance of SilK’s parallel implementation and computed that it will need an average of 4.36 days to finalize each task with 8 threads. Overall, our parallel implementation of Radon was up to 1725.77 times (834.69 times on average) faster than SilK, which clearly show the scalability of Radon’s parallel implementation.

In our fourth set of experiments, we aimed to compare Radon against Strabon on small datasets. The semantic spatio-temporal RDF store Strabon is not a LD framework but since it supports the GeoSPARQL and stSPARQL query languages, it can be employed for discovering topological relations via corresponding queries. To compare with Strabon, we used the same setting we used in the first set of experiments. Figure 3(b) shows the average runtimes result of both Radon and Strabon in seconds. In average, Radon was 11.99 times faster than Strabon. Interestingly, Strabon performed better than Radon on the intersects relation. The reason behind this behaviour is that Strabon uses an R-tree-over-GiST spatial index over the stored geometries in the underlying PostGIS database (Kyzirakos, Karpathiotakis, and Koubarakis 2012). This data structure is highly optimized for the retrieval of spatially connected objects. Hence, Strabon requires solely a data retrieval to compute.

Listing 1: SPARQL query for retrieving the intersects topological relation between resources from NUTS and CLC from Strabon.

\begin{verbatim}
SELECT ?s ?t WHERE {
  GRAPH <http://nuts.eu/> { ?s geo:asWKT ?s_geometry. }
  GRAPH <http://clc.eu/#243> { ?t geo:asWKT ?t_geometry. }
  FILTER( strdf:intersects(?s_geometry, ?t_geometry) )
}
\end{verbatim}
the \texttt{intersects} relation. However, this index is clearly outperformed by our sparse index in all the other relations.

In our \textit{fifth and last set of experiments}, we evaluated the scalability of \textsc{Radon} vs. \textsc{Strabon} when tackling big datasets. To this end, we applied the experimental setting we used in the second set of experiments ($S = T = \text{CLC}_m$). \textsc{Strabon} was not able to finish any of the experiments within the 2-hour time limit while \textsc{Radon} required approx. 95.10 minutes in the worst case. Given that \textsc{Strabon} provides no feedback pertaining to the progress of its tasks, we could not extrapolate its runtime. Thus, we attempted a smaller deduplication experiment with only one subset of CLC, \texttt{CLC-243}, which is about 10 times smaller than the merged \texttt{CLC}_m dataset. Even these experiments did not finish within the 2-hour limit. Therefore, we approximated \textsc{Strabon}’s runtime conservatively as follows: Assume that the \texttt{CLC-243} deduplication experiments would have finished just one minute after the 2-hour timeout. Assuming that \textsc{Strabon}’s runtime scales linearly with the input dataset size, the merged \texttt{CLC}_m experiments would take roughly 20.17 hours. Having this overly optimistic estimate of \textsc{Strabon}’s runtime, \textsc{Radon} achieves an average speedup of 24. When we move from the assumption that \textsc{Strabon} scales linearly to the more realistic assessment that it scales in $O(n^2)$, then we get an average speedup of 241. Overall, our results show clearly that \textsc{Radon} outperforms the state of the art by up to 3 orders of magnitude.

\section{Related Work}

Based on the original works of Egenhofer et al. (Egenhofer and Franzosa 1991), Clementini et al. (Clementini, Sharma, and Egenhofer 1994) propose the The DE-9IM model to capture the topological relations in $\mathbb{R}^2$. In addition, the \textit{Simple Features Model} proposed by OGC\footnote{http://www.opengeospatial.org/standards/sfs} contain different subsets of the topological relations that derive from the DE-9IM. GeoSPARQL (OGC 2010) is a recent OGC standard that proposes a query language that enable the discovery of topological relations. GeoSPARQL is implemented in the spatiotemporal RDF store \textsc{Strabon} (Kyzirokis, Karpathiotakis, and Koubarakis 2012). Other frameworks such as Virtuoso\footnote{http://virtuoso.openlinksw.com/} and newly BlazeGraph\footnote{http://www.blazegraph.com/} support geo-spatial extensions of SPARQL. The discovery of topological relations has been paid little attention to in previous research related to Link Discovery (Auer et al. 2013). Up to now, the state-of-the-art LD frameworks were able to discover only spatial similarities (Salas and Harth 2011; Sehgal, Getoor, and Viechnicki 2006; Vilches-Blázquez, Saquicela, and Corcho 2012). For example, (Ngonga Ngomo 2013) uses the \textit{Hausdorff} distance to compute the point-set distance between geo-spatial entities. In recent work, (Georganala, Sherif, and Ngonga Ngomo 2016) implements an efficient approach for \textit{Allen} Relations extraction. To the best of our knowledge, the only LD framework that support discovery of topological relations is \texttt{Silk} (Smeros and Koubarakis 2016). Based on \textit{MultiBlocking} technique, \texttt{Silk} computes the topological relations according to the DE-9IM standard between geo-spatial resources. A review of the current state of LD frameworks is in (Nentwig et al. 2015).

\section{Conclusions and Future Work}

We presented \textsc{Radon}, an approach for rapid discovery of topological relations among geo-spatial resources. \textsc{Radon} combines space tiling, minimum bounding box approximation and a sparse index to achieve a high scalability. We evaluated \textsc{Radon} with real datasets of various sizes and showed that in addition to being complete and correct, it also outperforms the state of the art by up to three orders of magnitude (e.g., \texttt{equals} relation against \tt{Silk}). In future work, we aim to apply more sophisticated load balancing approaches, such as the particle-swarm-optimization based approaches (Sherif and Ngomo 2015). In addition, we will consider the usage of other topology approximation methods.

\section*{Acknowledgments}

This work has been supported by the European Union’s H2020 research and innovation action HOBBIT (GA no. 688227) and the BMWI Project GEISER (project no. 01MD16014).
References


