RADON — Rapid Discovery of Topological Relations

Mohamed Ahmed Sherif,[a] Kevin Dreßler,[a] Panayiotis Smeros,[b] Axel-Cyrille Ngonga Ngomo[a]
[a] Department of Computer Science, University of Leipzig, 04109 Leipzig, Germany
{sherif|dressler|ngonga}@informatik.uni-leipzig.de
[b] 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 RADON – efficient solution for the discovery of topological relations between geospatial resources according to the DE9-IM 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.

[1] http://linkedgeodata.org

[2] http://nuts.geovocab.org/

[3] https://www.w3.org/DesignIssues/LinkedData.html

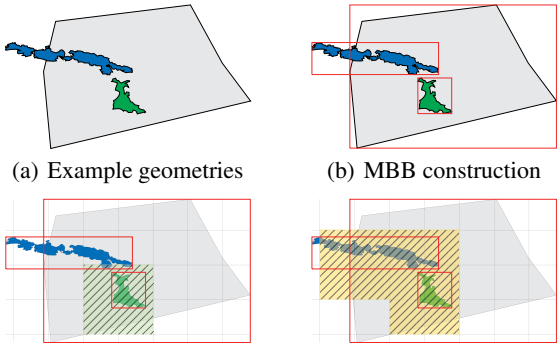
We go beyond the state of the art by providing *a novel indexing method combined with space tiling that allows for the efficient computation of topological relations between geo-spatial resources*. In particular, we present a novel sparse index for geo-spatial resources. We then develop a strategy to 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 $(s, p, o) \in (\mathcal{R} \cup \mathcal{B}) \times \mathcal{P} \times (\mathcal{R} \cup \mathcal{L} \cup \mathcal{B})$, where $\mathcal{R}$ is the set of all resources, $\mathcal{B}$ is the set of all blank nodes, $\mathcal{P}$ the set of all predicates and $\mathcal{L}$ 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

[4] Most commonly encoded in the WKT format, see http://www.opengeospatial.org/standards/sfa.



(a) Example geometries (b) MBB construction

(c) Space tiling (d) Optimized sparse space tiling

Figure 1: City of Leipzig from NUTS (in gray) together with topologically related geometries from CLC (in green and blue). See Section 4 for description of NUTS and CLC.

(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 3×3 intersection matrix:

$$DE9IM(a,b)\begin{bmatrix} dim(I(g_1) \cap I(g_2)) & dim(I(g_1) \cap B(g_2)) & dim(I(g_1) \cap E(g_2)) \\ dim(B(g_1) \cap I(g_2)) & dim(B(g_1) \cap B(g_2)) & dim(B(g_1) \cap E(g_2)) \\ dim(E(g_1) \cap I(g_2)) & dim(E(g_1) \cap B(g_2)) & dim(E(g_1) \cap E(g_2)) \end{bmatrix} \quad (1)$$

where *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 *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 $dim(x)$ with the binary domain $\{true, false\}$ is obtained using the boolean function $\beta(dim(I(g)) = false$ iff $dim(I(g)) = -1$ and *true* otherwise.

The major insight behind Radon 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 Radon 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 $\kappa_i(p)$ 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 $c_i^{\perp} = \min_{p \in g}\{\kappa_i(p)\}$ and the highest point coordinate $c_i^{\top} = \max_{p \in g}\{\kappa_i(p)\}$ in each dimension $i \in \{0, \ldots, n\}$. Then, the 2^n vertices of the MBB in n dimensions are all the vectors $\left(c_0^{(\cdot)}, c_1^{(\cdot)}, \ldots, c_n^{(\cdot)}\right)$, where $(\cdot) \in \{\perp, \top\}$. 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 Orchid (Ngonga Ngomo 2013) and $\mathcal{HR}^3$ (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 ℓ. These hypercubes are indexed with vectors $i \in \mathbb{N}^n$ 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 Δ_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 Radon algorithm (Algorithm 1). Radon takes a set of source resources S, a set of target resources T and a topological relation r as input. The goal of Radon is to generate the mapping $M = \{(s,t) \in S \times T : r(s,t)\}$ *efficiently*, where r is a topological relation. Radon 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

$$\text{eth}(X) = |X| \prod_{i=1}^{d} \frac{1}{|X|} \sum_{x \in X} \left(\max_{p \in x}\{\kappa_i(p)\} - \min_{p \in x}\{\kappa_i(p)\} \right), \quad (2)$$

with d being the number of dimensions of the resource geometries and $\kappa_i(p)$ denoting the coordinate of a point p in the i^{th} dimension. If $\text{eth}(T) < \text{eth}(S)$, Radon swaps S and T and computes the reverse[5] 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 Radon swaps T and S 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

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

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 ETH of NUTS (containing only the gray geometry) is greater than the ETH of CLC (containing the green and blue areas). Thus, we set S = CLC and T = NUTS.

3.2 Optimized Sparse Space Tiling

In its second step, 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 Δ_φ and Δ_λ 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 $(\varphi_1(x), \lambda_1(x))$ and $(\varphi_2(x), \lambda_2(x))$ 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 $\psi_\perp$ and $\psi_\top$ from Equation 3.

$$\psi^\perp(x) = \lfloor x \cdot \Delta_\varphi \rfloor \qquad \psi^\top(x) = \lceil x \cdot \Delta_\varphi \rceil \tag{3}$$

We then map x to all hypercubes with indices (i, j) where $i, j \in \mathbb{Z}$, $\psi^\perp(\varphi_1(x)) \leq i \leq \psi^\top(\varphi_2(x))$ and $\psi^\perp(\lambda_1(x)) \leq j \leq \psi^\top(\lambda_2(x))$. 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 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 $\Delta_\varphi = \Delta_\lambda = 2$. The green area's MBB has the following corner coordinates: $(\varphi_1(g), \lambda_1(g))$ = (12.340703846780286, 51.28797110806819) and $(\varphi_2(g), \lambda_2(g))$ = (12.389192648396918, 51.33902633403139). Therefore, $\psi^\perp(\varphi_1(g)) = 24$, $\psi^\perp(\lambda_1(g)) = 102$, $\psi^\top(\varphi_2(g)) = 25$, $\psi^\top(\lambda_2(g)) = 103$ 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, 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), RADON first discards unnecessary computations using the TESTMBB procedure. TESTMBB 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. `equals`, `covers`, `within` formally defined in the annex (Sherif et al. 2016)). Let $\square(g)$ denote the MBB geometry of a geometry g. Note that $g \subseteq \square(g)$ always holds. We can now infer $\neg r(\square(s), \square(t)) \Rightarrow \neg r(s, t)$ using the transitivity of $\subseteq$. For all other relations, TESTMBB simply returns *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 TESTMBB method returns true, RADON carries out the more expensive computation of the topological relation between the geometries s and t (Line 31). If $r(s, t)$ holds, 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. RADON *is complete and correct w.r.t. the application of space tiling.*[6]

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 `equals`, `within`, `contains`, `disjoint`, `touches`, `meets`, `covers`, `coveredBy`, `intersects`, `inside`, `crosses` and `overlaps`. Note that some of these relations are *synonyms* (e.g., $\texttt{touches}(x, y) \Leftrightarrow \texttt{meets}(x, y)$) while others are *combinations* of more atomic relations,(e.g., $\texttt{equals}(x, y) \Leftrightarrow \texttt{within}(x, y) \wedge \texttt{contains}(x, y)$). Moreover, some relations are the *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 *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 RADON against two state-of-the-art approaches; (1) SILK as it is (to the best of our knowledge) the only LD framework that supports the discovery of topological relation, (2) STRABON as it implements the standard GeoSPARQL (OGC 2010) and is based on PostGIS. For SILK experiments, we ran our experiments using its latest version (v2.6.1) with a blocking factor of 10 as in (Smeros and Koubarakis 2016). For 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

```
input : S, set of source resources. T, set of target
        resources. r, topological relation.
output: M, Mapping {(s,t) ∈ S × T : r(s,t)}
reversed ← false;
if ETH(T) < ETH(S) then
    SWAP(S, T);
    r ← r';
    reversed ← true;
/* Get index I using optimized sparse
   space tiling                                   */
(Δφ, Δλ) ← FINDBESTGRANULARITY(S, T);
foreach geometry s ∈ S do
    (φ1(s), λ1(s), φ2(s), λ2(s)) ←
      GETMBBDIAGONALCORNERS(s);
    for i ← ⌊φ1(s) · Δφ⌋ to ⌈φ2(s) · Δφ⌉ do
        for j ← ⌊λ1(s) · Δλ⌋ to ⌈λ2(s) · Δλ⌉ do
            INSERTINTOHYPERCUBE(I(S), i, j, s);
            j ← j + 1;
        i ← i + 1;
foreach geometry t ∈ T do
    (φ1(t), λ1(t), φ2(t), λ2(t)) ←
      GETMBBDIAGONALCORNERS(t);
    for i ← ⌊φ1(t) · Δφ⌋ to ⌈φ2(t) · Δφ⌉ do
        for j ← ⌊λ1(t) · Δλ⌋ to ⌈λ2(t) · Δλ⌉ do
            if GETHYPERCUBE(I(S), i, j) is not empty
              then
                INSERTINTOHYPERCUBE(I(T), i, j, t);
            j ← j + 1;
        i ← i + 1;
/* Generate Links                                 */
foreach hypercube H_S ∈ I(S) do
    H_T ← GETHYPERCUBE(I(T), φ(H_S), λ(H_S));
    if H_T is not empty then
        for s ∈ H_S do
            for t ∈ H_T do
                if (s,t) ∉ C then
                    C ← C ∪ {(s,t)};
                    B ← (φ1(s), λ1(s), φ2(s), λ2(s)),
                      (φ1(t), λ1(t), φ2(t), λ2(t)) ;
                    if TESTMBB(r, B) then
                        if r(s,t) is true then
                            M ← M ∪ {(s,t)};
if reversed then return M' ;
else 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 completeness 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` serialisation 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* Δ_φ and Δ_λ used for tiling the space and generating the sparse index of hypercubes. We tried 4 different heuristics corresponding to a statistical measure: *minimum, maximum, median* and *average*. Each heuristic first computes the respective statistical measure η independently for both datasets and both dimensions, resulting in 4 temporary values $h_{\eta,\varphi}(S)$, $h_{\eta,\varphi}(T)$, $h_{\eta,\lambda}(S)$, $h_{\eta,\lambda}(T)$. Finally, the granularity factor in *each dimension* is the average of the two datasets. Formally,

$$h_{\eta,\varphi}(X) = \underset{x \in X}{\eta} \left\{ \max_{p \in x}\{\varphi(p)\} - \min_{p \in x}\{\varphi(p)\} \right\} \tag{4}$$

$$\Delta_{\eta,\varphi}(S,T) = \frac{1}{2}\Big(h_{\eta,\varphi}(S) + h_{\eta,\varphi}(T)\Big) \tag{5}$$

$h_{\eta,\lambda}(X)$ 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 $\eta \in \{min, max, avg, median\}$. We used all the 44 subsets of the CLC dataset as input for this experiment and recorded how many times each heuristic achieved the best runtime for the `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 *average* heuristic achieves the best result 24 times out of 44 experiments. Runner-up is *median*, achieving the best runtime 17 times. Finally, the *min* and *max* heuristics achieved only 2 and 1 time(s) respectively. Interestingly, *average* and *median* were only 4% slower than the best measure on average when not being the best, while *min* and *max* where 34%

[8] http://aksw.org/Projects/LIMES

[9] Version 0.91 (http://nuts.geovocab.org/data/0.91/) is used in this work.

[10] For more details about CLC see https://datahub.io/dataset/corine-land-cover

[11] https://github.com/silk-framework/silk/issues/57

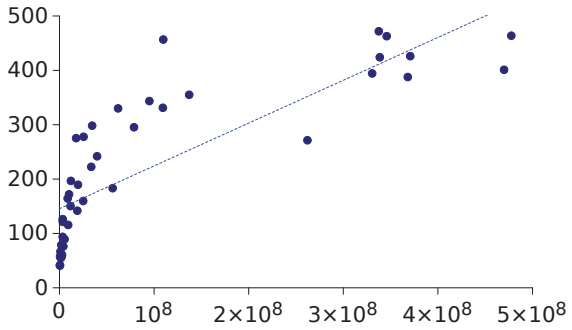


Figure 2: Speedup of RADON over SILK for the `within` relation. The x-axis represents the dataset sizes, y-axis represents the speedup. The blue line is the linear regression line.

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 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 ran 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_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

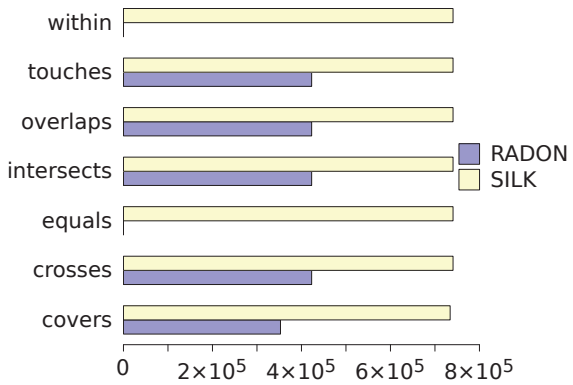
```
SELECT ?s ?t WHERE {
  GRAPH <http://nuts.eu/>    {?s geo:asWKT ?s_geometry.}
  GRAPH <http://clc.eu/#243> {?t geo:asWKT ?t_geometry.}
  FILTER( strdf:intersects(?s_geometry, ?t_geometry) )
}
```

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

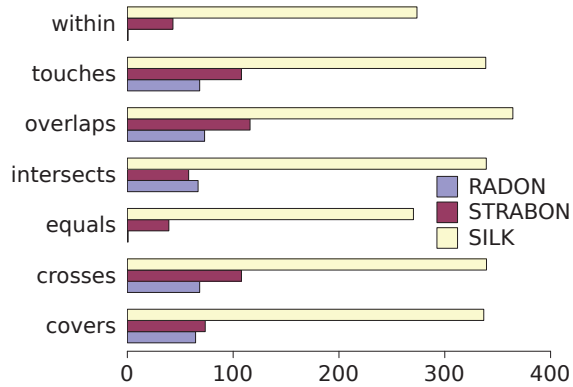
average (maximum = 95.10 minutes for the `crosses` relation). SILK was only able to (on average) finalize 0.34% of each task within the 2 hour timeout limit. We extrapolated the runtime of SILK linearly to get an approximation of how long it would need to carry out the tasks at hand. On average, SILK would need 24.85 days to complete each task (linear extrapolation). Consequently, RADON is at least 715.16 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_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 `intersect` 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



(a) Average number of computations of topological relations



(b) Average runtime

Figure 3: Average number of complete computations of topological relations and average runtime for the datasets experiments. All runtimes are in seconds.

the `intersects` relation. However, this index is clearly outperformed by our sparse index in all the other relations.

In our *fifth and last set of experiments*, we evaluated the scalability of RADON vs. STRABON when tackling big datasets. To this end, we applied the experimental setting we used in the second set of experiments ($S = T = CLC_m$). STRABON was not able to finish any of the experiments within the 2-hour time limit while RADON required approx. 95.10 minutes in the worst case. Given that 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, CLC-243, which is about 10 times smaller than the merged CLC_m dataset. Even these experiments did not finish within the 2-hour limit. Therefore, we approximated STRABON's runtime conservatively as follows: Assume that the CLC-243 deduplication experiments would have finished just one minute after the 2-hour timeout. Assuming that STRABON's runtime scales linear with the input dataset size, the merged CLC_m experiments would take roughly 20.17 hours. Having this overly optimistic estimate of STRABON's runtime, RADON achieves an average speedup of 24. When we move from the assumption that 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 RADON outperforms the state of the art by up to 3 orders of magnitude.

5 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 the $\mathbb{R}^2$. In addition, the *Simple Features Model* proposed by OGC[12] 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 STRABON (Kyzirakos, Karpathiotakis, and Koubarakis 2012). Other frameworks such as Virtuoso[13] and newly BlazeGraph[14] 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 *Hausdorff* distance to compute the point-set distance between geo-spatial entities. In recent work, (Georgala, Sherif, and Ngonga Ngomo 2016) implements an efficient approach for *Allen* Relations extraction.To the best of our knowledge, the only LD framework that support discovery of topological relations is SILK (Smeros and Koubarakis 2016). Based on *MultiBlocking* technique, 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).

6 Conclusions and Future Work

We presented RADON, an approach for rapid discovery of topological relations among geo-spatial resources. RADON combines space tiling, minimum bounding box approximation and a sparse index to achieve a high scalability. We evaluated 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., `equals` relation against 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.

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).

[12] http://www.opengeospatial.org/standards/sfs
[13] http://virtuoso.openlinksw.com/
[14] https://www.blazegraph.com/

References

Auer, S.; Lehmann, J.; Ngomo, A.-C. N.; and Zaveri, A. 2013. Introduction to Linked Data and Its Lifecycle on the Web. In Rudolph, S.; Gottlob, G.; Horrocks, I.; and van Harmelen, F., eds., *Reasoning Web*, volume 8067 of *LNCS*, 1–90. Springer.

Clementini, E.; Di Felice, P.; and van Oosterom, P. 1993. *Advances in Spatial Databases: 3rd International Symposium, SSD '93 Singapore, June*. Springer. chapter A small set of formal topological relationships suitable for end-user interaction, 277–295.

Clementini, E.; Sharma, J.; and Egenhofer, M. J. 1994. Modelling topological spatial relations: Strategies for query processing. *Computers & graphics* 18(6):815–822.

Egenhofer, M. J., and Franzosa, R. D. 1991. Point-set topological spatial relations. *International Journal of Geographical Information System* 5(2):161–174.

Georgala, K.; Sherif, M. A.; and Ngonga Ngomo, A.-C. 2016. An Efficient Approach for the Generation of Allen Relations. In *European Conference on Artificial Intelligence (ECAI)*.

Kyzirakos, K.; Karpathiotakis, M.; and Koubarakis, M. 2012. Strabon: A semantic geospatial DBMS. In Cudré-Mauroux, P.; Heflin, J.; Sirin, E.; Tudorache, T.; Euzenat, J.; Hauswirth, M.; Parreira, J. X.; Hendler, J.; Schreiber, G.; Bernstein, A.; and Blomqvist, E., eds., *ISWC 2012, Boston, USA, November, 2012*, volume 7649 of *LNCS*, 295–311. Springer.

Lehmann, J.; Furche, T.; Grasso, G.; Ngonga Ngomo, A.-C.; Schallhart, C.; Sellers, A.; Unger, C.; Bühmann, L.; Gerber, D.; Höffner, K.; Liu, D.; and Auer, S. 2012. Deqa: Deep web extraction for question answering. In *Proceedings of ISWC*.

Lehmann, J. 2009. DL-Learner: learning concepts in description logics. *Journal of Machine Learning Research (JMLR)* 10:2639–2642.

Nentwig, M.; Hartung, M.; Ngomo, A.-C. N.; and Rahm, E. 2015. A survey of current link discovery frameworks. *Semantic Web Journal*.

Ngonga Ngomo, A.-C. 2012. Link discovery with guaranteed reduction ratio in affine spaces with minkowski measures. In *International Semantic Web Conference (1)*, 378–393.

Ngonga Ngomo, A.-C. 2013. Orchid–reduction-ratio-optimal computation of geo-spatial distances for link discovery. In *The Semantic Web–ISWC 2013*. Springer. 395–410.

OGC. 2010. GeoSPARQL - A geographic query language for RDF data.

O'Rourke, J. 1985. Finding minimal enclosing boxes. *International journal of computer & information sciences* 14(3):183–199.

Salas, J., and Harth, A. 2011. Finding spatial equivalences accross multiple RDF datasets. In *Proceedings of the Terra Cognita Workshop on Foundations, Technologies and Applications of the Geospatial Web*, 114–126. Citeseer.

Sehgal, V.; Getoor, L.; and Viechnicki, P. D. 2006. Entity resolution in geospatial data integration. In *Proceedings of the 14th annual ACM international symposium on Advances in geographic information systems*, 83–90. ACM.

Sherif, M. A., and Ngomo, A.-C. N. 2015. An optimization approach for load balancing in parallel link discovery. In *Proceedings of the 11th International Conference on Semantic Systems*, SEMANTICS '15, 161–168. New York, NY, USA: ACM.

Sherif, M. A.; Dreßler, K.; Smeros, P.; and Ngonga Ngomo, A.-C. 2016. Annex: Radon - Rapid Discovery of Topological Relations. *ArXiv e-prints*.

Shreedhar, M., and Varghese, G. 1996. Efficient fair queuing using deficit round-robin. *Networking, IEEE/ACM Transactions on* 4(3):375–385.

Smeros, P., and Koubarakis, M. 2016. Discovering Spatial and Temporal Links Among RDF Data. In *WWW2016 Workshop: Linked Data on the Web (LDOW2016)*.

Vilches-Blázquez, L. M.; Saquicela, V.; and Corcho, O. 2012. Interlinking geospatial information in the web of data. In *Bridging the Geographic Information Sciences*. Springer. 119–139.