Benchmarking Link Discovery Systems for Geo-Spatial Data*

Tzanina Saveta[1], Giorgos Flouris[1], Irini Fundulaki[1], and
Axel-Cyrille Ngonga Ngomo[2]

[1] Institute of Computer Science-FORTH Greece,
[2] Paderborn University

Abstract. Linking geo-spatial entities is targeted only by a limited number of *link discovery benchmarks*. Linking spatial resources requires techniques that differ from the classical, mostly string-based approaches. In particular, considering the topology of the spatial resources and the topological relations between them is of central importance to systems that manage spatial data. Due to the large amount of available geo-spatial Linked Data datasets, it is critical that benchmarks for geo-spatial link discovery systems are developed that can determine the effectiveness and the efficiency of the proposed techniques. In this paper, we propose a *Spatial Benchmark* generator that deals with link discovery for spatial data. Our benchmark generator can be used to test the performance of systems that deal with all the topological relations proposed in the state of the art DE-9IM (Dimensionally Extended nine-Intersection) model in the two dimensional space. We also provide a comparative analysis with benchmarks produced using the *Spatial Benchmark* generator to assess and identify the capabilities of RADON and Silk, two of the state of the art systems.

1 Introduction

The number of datasets published in the Web of Data as part of the Linked Data Cloud is constantly increasing. The Linked Data paradigm is based on the unconstrained publication of information by different publishers, and the interlinking of Web resources across knowledge bases. In most cases, the cross-dataset links are not explicit in the dataset and must be automatically determined using *link discovery* tools amongst others [1]. The large variety of link discovery techniques requires their comparative evaluation to determine which one is best suited for a given context. Performing such an assessment generally requires well-defined and widely accepted *benchmarks* to determine the weak and strong points of the proposed techniques and/or tools.

A number of real and synthetic benchmarks that address different data challenges have been proposed for evaluating the performance of such systems [2]. However, to the best of our knowledge, there is no benchmark to target the problem of linking geo-spatial entities. This is in contrast to the fact that, some

* The presented work was funded by the H2020 project HOBBIT (#688227).

of the largest knowledge bases on the Linked Open Data Web are geo-spatial knowledge bases (e.g., LinkedGeoData,[1] with more than 30 billion triples).

Linking spatial resources requires techniques that differ from the classical mostly string-based approaches. In particular, considering the topology of the spatial resources and the topological relations between them is of central importance to systems that manage spatial data. We believe that due to the large amount of available geo-spatial datasets employed in Linked Data and in several domains, it is critical that benchmarks for geo-spatial link discovery are developed.

The present paper proposes the *Spatial Benchmark* generator[2] that deals with Link Discovery for spatial data. The benchmark can be used to test the performance of systems that deal with topological relations proposed in the state of the art DE-9IM (Dimensionally Extended nine-Intersection Model) model [3]. This benchmark generator implements all topological relations of DE-9IM between *trajectories* in the *two-dimensional space*. In our work, we focus on (a) the correct implementation of the DE-9IM relations by the systems and (b) scalability, more specifically, check if the systems are able to handle large datasets properly.

To the best of our knowledge such a generic benchmark, that takes as input trajectories and checks the performance of linking systems for spatial data does not exist. We also provide a comparative analysis with benchmarks produced using the *Spatial Benchmark* generator to assess and identify the capabilities of RADON [4] and Silk [5], two of the state-of-the-art systems.

2 Dataset and Ontology

The Spatial Benchmark is based on TomTom datasets[3] provided in the context of the HOBBIT H2020 project[4]. TomTom's traffic data archive contains approximately 15 trillion $(1.5 \cdot 10^{13})$ speed measurements from hundreds of millions of road segments all over the world, mostly based on anonymized GPS positioning data collected with user's consent. This data has been used to extract the recurrent traffic patterns and adds this to digital navigation maps as a map layer. Statistical traffic data also form an important input set for TomTom's live traffic fusion engine. Probe data is being used for custom-made traffic and demand analysis. Furthermore, archived positioning data is used to improve digital maps, road geometry and map attributes.

The ontology we use to represent TomTom's data is shown in Figure 1. The main class is `Trace` that contains one or more points (class `Point`). Class `Point` is a subclass of `wgs84_pos:Point` class of the WGS84 Geo Positioning (geo)[5] vocabulary that represents latitude, longitude and altitude information

[1] `http://linkedgeodata.org/About`

[2] `https://github.com/hobbit-project/SpatialBenchmark`

[3] `https://www.tomtom.com/en_gr/`

[4] `http://project_hobbit.eu`

[5] `http://lov.okfn.org/dataset/lov/vocabs/geo`

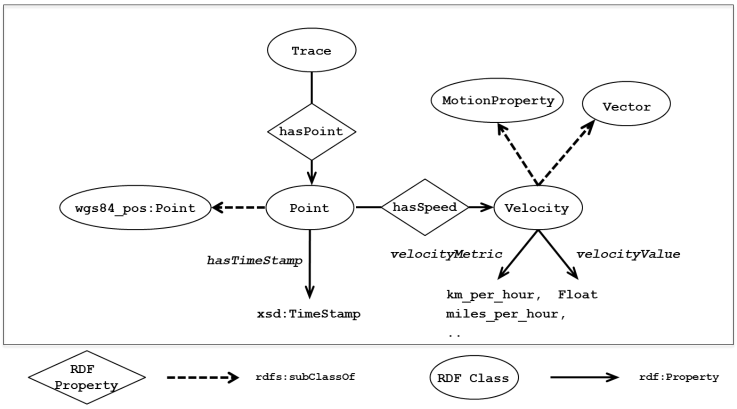


Fig. 1: TomTom Ontology

in the WGS84 geodetic reference datum. Each point is associated with a *velocity* (class `Velocity`), instances of which have properties `velocityMetric` and `velocityValue`, the former taking values from a predefined set (*km_per_hour*, *miles_per_hour*) and the later from class `xsd:Float`. A point also has attribute `hasTimeStamp` that takes its values in class `xsd:TimeStamp` which designates the time an object was at this specific point. Listing 1.1 shows an example of a Trace consisting of 4 Points.

```
@base <http://www.tomtom.com/trace-data/0000000001.ttl> .
@prefix : <http://www.tomtom.com/ontologies/traces#> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .

<#trace> a :Trace .
<#trace> :hasPoint <#point0> .
<#point0> :hasTimestamp "2010-03-12T10:19:00.000000"^^xsd:dateTime ;
      :lat 40.898320 ;
      :long 14.185080 ;
      :hasSpeed <#speed0> .
<#speed0> :velocityValue 7.22 ;
      :velocityMetric :kilometers_perHour .
<#trace> :hasPoint <#point1> .
<#point1> :hasTimestamp "2010-03-12T10:19:10.000000"^^xsd:dateTime ;
      :lat 40.898560 ;
      :long 14.184440 ;
      :hasSpeed <#speed1> .
<#speed1> :velocityValue 6.67 ;
      :velocityMetric :kilometers_perHour .
<#trace> :hasPoint <#point2> .
<#point2> :hasTimestamp "2010-03-12T10:19:20.000000"^^xsd:dateTime ;
      :lat 40.898950 ;
      :long 14.184130 ;
      :hasSpeed <#speed2> .
<#speed2> :velocityValue 5.28 ;
      :velocityMetric :kilometers_perHour .
<#trace> :hasPoint <#point3> .
<#point3> :hasTimestamp "2010-03-12T10:19:30.000000"^^xsd:dateTime ;
      :lat 40.899410 ;
      :long 14.184080 ;
      :hasSpeed <#speed3> .
<#speed3> :velocityValue 5.28 ;
      :velocityMetric :kilometers_perHour .
```

Listing 1.1: TomTom Data

3 Spatial Benchmark

3.1 Dimensionally Extended nine-Intersection Model (DE-9IM)

The Dimensionally Extended nine-Intersection Model (DE-9IM) [3] is a topological model and a standard, used to describe the spatial relations of two geometries in two-dimensional space. The DE-9IM model is based on a 3 × 3 *Intersection Matrix* of the form:

$$\text{DE9IM}(a,b) = \begin{bmatrix} \dim(I(a) \cap I(b)) & \dim(I(a) \cap B(b)) & \dim(I(a) \cap E(b)) \\ \dim(B(a) \cap I(b)) & \dim(B(a) \cap B(b)) & \dim(B(a) \cap E(b)) \\ \dim(E(a) \cap I(b)) & \dim(E(a) \cap B(b)) & \dim(E(a) \cap E(b)) \end{bmatrix}$$

where dim is the maximum number of dimensions of the intersection ($\cap$) of the interior (I), boundary (B), and exterior (E) of geometries g_1 and g_2. Dimension ($\dim(x)$) values are obtained mapping the value 1 (for lines) to T (true), so using the boolean domain $\{T, F\}$. The supported spatial relations of DE-9IM are formally described below (see also Figure 2). Here, we will write $(I(g_1)I(g_2))$ to show $\dim(I(g_1) \cap I(g_2))$ etc.

- **Equal**: Two geometries g_1 and g_2 are **topologically equal** if their interiors intersect and no part of the interior or boundary of one geometry intersects the exterior of the other. *Topologically equal* is equivalent to *Within* and *Contains* DE-9IM relations. Formally:

$$(I(g_1)I(g_2)) \wedge \neg(I(g_1)E(g_2)) \wedge \neg(B(g_1)E(g_2)) \wedge \neg(E(g_1)I(g_2)) \wedge \neg(E(g_1)B(g_2))$$

- **Disjoint**: Two geometries g_1 and g_2 are **disjoint** if they have no point in common. They form a set of disconnected geometries. *Disjoint* is equivalent to *not Intersects.* Formally:

$$\neg(I(g_1)I(g_2)) \wedge \neg(I(g_1)B(g_2)) \wedge \neg(B(g_1)I(g_2)) \wedge \neg(B(g_1)B(g_2))$$

- **Touches**: A geometry g_1 **touches(meets)** a geometry g_2 if they have *at least one boundary point in common, but no common interior points.* Formally:

$$(\neg(I(g_1)I(g_2)) \wedge I(g_1)B(g_2)) \vee (\neg(I(g_1)I(g_2)) \wedge B(g_1)I(g_2)) \vee \\ (\neg(I(g_1)I(g_2)) \wedge B(g_1)B(g_2))$$

- **Contains:** A geometry g_1 **contains** a geometry g_2 if g_2 lies in g_1, and the interiors of the two geometries intersect. *Contains* is equivalent to *Within(g_2, g_1).* Formally:

$$(I(g_1)I(g_2)) \wedge \neg(E(g_1)I(g_2)) \wedge \neg(E(g_1)B(g_2))$$

- **Covers:** A geometry g_1 **covers** a geometry g_2 if geometry g_2 lies in g_1. Other definitions: "no points of g_2 lie in the exterior of g_1", or "Every point of g_2 is a point of (the interior or boundary of) g_1". *Covers* is equivalent to *CoveredBy(g_2, g_1)*. Formally:

$$((I(g_1)I(g_2)) \wedge \neg(E(g_1)I(g_2)) \wedge \neg(E(g_1)B(g_2)))\vee$$
$$((I(g_1)B(g_2)) \wedge \neg(E(g_1)I(g_2)) \wedge \neg(E(g_1)B(g_2)))\vee$$
$$((B(g_1)I(g_2)) \wedge \neg(E(g_1)I(g_2)) \wedge \neg(E(g_1)B(g_2)))\vee$$
$$((B(g_1)B(g_2)) \wedge \neg(E(g_1)I(g_2)) \wedge \neg(E(g_1)B(g_2)))$$

- **Intersects:** A geometry g_1 **intersects** g_2 if they have at least one point in common. *Intersects* is equivalent to *not Disjoint.*
- **Within:** A geometry g_1 is **within(inside)** g_2 if g_1 lies in the interior of g_2. *Within* is equivalent to *Contains(g_2, g_1).*
- **Covered by:** A geometry g_1 is **covered by** g_2 (extends *Within*) if every point of g_1 is a point of g_2, and the interiors of the two geometries have at least one point in common. *Covered by* is equivalent to *Covers(g_2, g_1).*
- **Crosses:** A geometry g_1 **crosses** g_2 if they have some but not all interior points in common, and the dimension of the intersection is less than that of at least one of them. Mask selection rules are checked only when $\dim(g_1) \neq \dim(g_2)$, except by line/line inputs, otherwise is false:

$$ll((I(g_1)I(g_2)) = 0) \text{ for lines}$$
$$(I(g_1)I(g_2) \wedge I(g_1)E(g_2)) \text{ when } \dim(g_1) < \dim(g_2)$$
$$(I(g_1)I(g_2) \wedge E(g_1)I(g_2)) \text{ when } \dim(g_1) > \dim(g_2)$$

- **Overlaps:** A geometry g_1 is **overlaps** g_2 if they have some but not all points in common, they have the same dimension, and the intersection of the interiors of the two geometries has the same dimension as the geometries themselves. Mask selection rules are checked only when $\dim(g_1) = \dim(g_2)$, otherwise is false:

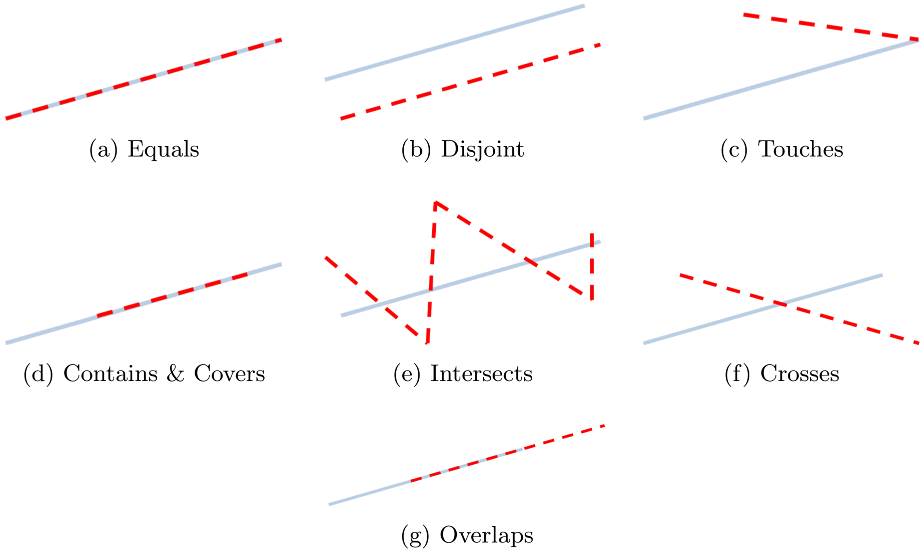
$$ll((I(g_1)I(g_2)) \wedge (I(g_1)E(g_2)) \wedge (E(g_1)I(g_2))) \text{ for points or surfaces}$$
$$((I(g_1)I(g_2)) = 1 \wedge (I(g_1)E(g_2)) \wedge (E(g_1)I(g_2))) \text{ for lines}$$

Examples of the relations are shown in Figure 2.

3.2 Overview

For the Spatial Benchmark, we focus on transformations that follow the DE-9IM (Dimensionally Extended nine-Intersection Model) topological relations and determine whether the systems are able to identify such relations.

For the design of this benchmark, we focused *(a)* on the correct implementation of all the topological relations of the DE-9IM topological model and *(b)* on



(a) Equals

(b) Disjoint

(c) Touches

(d) Contains & Covers

(e) Intersects

(f) Crosses

(g) Overlaps

Fig. 2: Examples of topological relations

producing datasets large enough to stress the systems under test. To the best of our knowledge, there exist few systems that implement the topological relations of DE-9IM, hence the benchmark already addresses the first choke point set. Moreover, we produced very large synthetic datasets using TomTom's original data, and hence we are able to challenge the systems regarding scalabity.

The benchmark gets as input a *set of traces*, each *trace* being a *sequence of points*. The points are expressed using *standard longitude/latitude coordinates*. We preprocess the datasets to represent them in the Well-known text (WKT) format[6], widely used by systems that manage geo-spatial data. WKT is a text markup language for representing vector geometry objects on a map, spatial reference systems of spatial objects and transformations among spatial reference systems. WKT offers a compact machine and human readable representation of geometric objects. Nevertheless, if the input dataset comes in a different format, then we can easily transform the data in the required format. Appropriate transformations are applied to the input set of traces in order to obtain the target dataset that can be used to test the ability of systems to identify DE-9IM topological relations.

More specifically, as shown in Figure 3, the benchmark gets as input traces represented as *LineStrings*, with a linestring being a one-dimensional object representing a sequence of points and the line segments connecting them, and produces a *source* and a *target* dataset. The source dataset is identical to the input dataset, whereas the target dataset is generated in such a way so that its traces have specific topological (DE-9IM) relations with the traces of the source

[6] `https://en.wikipedia.org/wiki/Well-known_text`

dataset. The gold standard is produced after the generation of the source and target datasets.

The benchmark generator makes use of the *Initialization*, *Resource Generator* and the *Resource Transformation* modules:

1. The *Initialization Module* reads the test case generation parameters and retrieves by means of SPARQL queries each trace instance.
2. The *Resource Generator* uses the trace instances retrieved from the Initialization Module for the generation of the *source dataset*.
3. The *Resource Transformation Module* uses the JTS Topology Suite[7] to return for each source instance s_i the transformed instance t_i that has a certain predecided relation (from the DE-9IM set) with s_i.

Note that Step **3** above may lead to the creation of certain t_i that inadvertently have some relation with some s_j $(i \neq j)$. To address this problem, the *gold standard* is produced using RADON, in order to guarantee that all existing topological relations (intended or not) will be included in the gold standard.

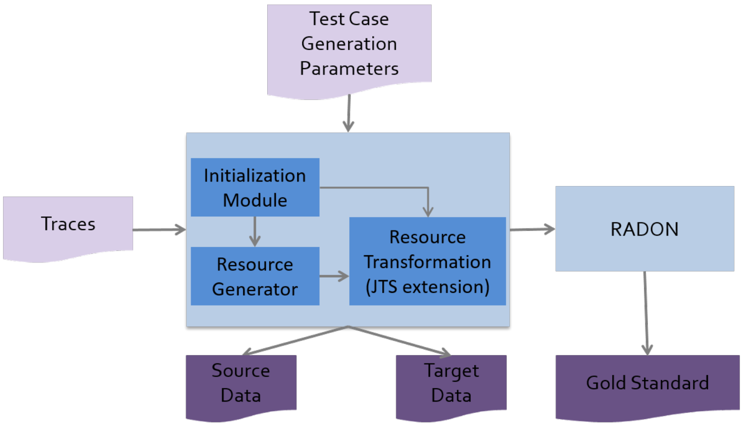


Fig. 3: Spatial benchmark architecture

3.3 Benchmark Parameters

For the generation of the source and target datasets, the user has to provide values for the following benchmark parameters:

1. Number of instances to retrieve from the input dataset.
2. Percentage of points to keep for each Trace in the input dataset (this is due to the large number of points in each trace).
3. DE-9IM topological relation that will be tested for the source and target traces. As we mentioned earlier, a target trace t_i is likely to have a relation with s_j $(i \neq j)$ in addition to the source trace s_i from which it is generated.

[7] https://en.wikipedia.org/wiki/JTS_Topology_Suite

3.4 Source and Target Data Generation

Source Dataset In order to generate the *source* dataset we proceed as follows: from the input dataset, we only keep longitude and latitude information from each point. With those points, we create, for each trace, a LineString[8] that is represented in the **Well-known text (WKT)** representation format.

Target Dataset In order to produce the *target* dataset, we retrieve source instances in a sequential manner, and using the Extension of the JTS Topology Suite we generate one target instance for each source instance. The target instance is defined in such a way so as the two instances (source and target) have the DE-9IM relation that was specified in the benchmark's configuration parameters. This process is continued until the requested number of instances is produced.

The **JTS Topology Suite**[9] is a Java API that implements a core set of spatial data operations using an explicit precision model and robust geometric algorithms. It provides a complete model for specifying 2D linear geometries. Many common operations in computational geometry and spatial data processing are exposed in a clear, consistent and integrated API in JTS. JTS is intended to be used in the development of applications that support the validation, cleaning, integration and querying of spatial datasets. JTS is based on the notion of the *bounding box (bbox)*, which is an area defined by two longitudes (in the range $-180 \dots 180$) and two latitudes (in the range $-90 \dots 90$), such that the resulting box (included within these coordinates) is the minimum box that contains the geometry under study.

For generating the target data in the Spatial Benchmark, we decided to implement an extension[10] of the JTS Topology Suite that allows one to generate a target geometry, given a source geometry and the intended Intersection Matrix (i.e., DE-9IM relation) between the source and target geometries. Below, we describe the algorithm of the implemented extension for each target relation of DE-9IM. As we will see, all cases are essentially based on the algorithms that generate *disjoint LineStrings*, along with some random selection of points and/or series of points. More specifically:

1. equal: Given a source LineString, an equal LineString is the exact same.

2. disjoint: Given a source LineString s we determine its bbox ($b(s)$), and randomly define coordinates for a bbox (say b') that does not intersect $b(s)$. This is done by just taking sufficiently large (or sufficiently small) coordinates for the minimum (maximum) longitude or latitude coordinates. Finally, we generate a random LineString that entirely falls inside b', thereby guaranteeing disjointness.

[8] A **LineString** is a WKT geometric object and consists of a sequence of two or more vertices, along with all points of the linearly interpolated curves (line segments) between each pair of consecutive vertices. The line segments in the line may intersect with each other (in other words, the linestring may "curl back" in itself and self-intersect). A linestring must have either zero or two or more points.

[9] `https://svn.code.sf.net/p/jts-topo-suite/code/tags/Version_1.14`

[10] `https://github.com/jsaveta/jtsExtension`

In rare cases, it could happen that $b(s)$ covers the entire plane, so no b' can be defined. In these cases, we randomly break the original LineString into several smaller LineStrings (say $s_1, \ldots, s_k$), and compute their bboxes ($b(s_1), \ldots, b(s_k)$). Then, we use the above process to identify a bbox b' that does not intersect with any of them and create a random target LineString as above.

If, despite the partitioning of s, no appropriate b' can be found, we define an alternative, more fine-grained partition and repeat the process which ends when an appropriate b' is found, or when each pair of consecutive points of s is a partition; if even this fine-grained partition does not allow the definition of an appropriate bbox, then the original LineString covers the entire plane and no disjoint line can be created.

An example of this partition, is shown in Figure 4. In the first subfigure, the bbox of the LineString covers the entire plane, thus we partitioned the LineString into 3 smaller parts. Then, we computed the new smaller bounding boxes and as is shown in the second image, there is now space (blue area) where we can define a new bbox and create the new LineString in it. We follow the same approach for the rest of the relations.

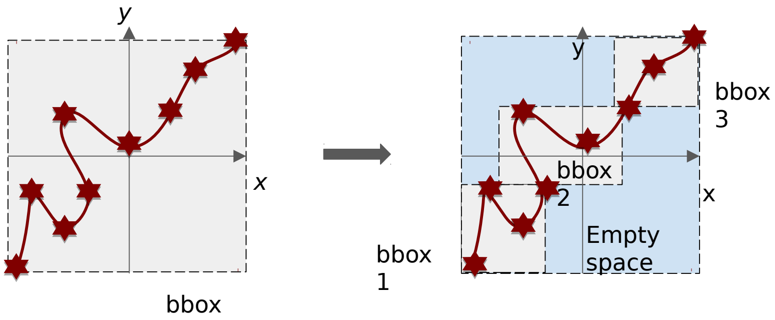


Fig. 4: Partitioning of a bbox example

3. contains & covers: Given a source LineString s, we randomly partition it into $s_1, \ldots, s_n$, and randomly pick one s_i to be the target LineString t.

4. intersects: Given a source LineString s, we randomly pick n points $p_1, \ldots, p_n$ ($n > 1$). Then, we create disjoint LineString $p_1, \ldots, p_{k-n}$, where k the size of LineString s. Finally, we create the final LineString t by adjoining the LineString t and the n points.

5. within & covered by: Within and covered by are obtained from contains and covers respectively (they are their inverse relations).

6. crosses: This is the same as intersects, except that $n = 1$, and that the chosen point must be an intermediate one.

7. overlaps: Given a source LineString s, we pick an intermediate point, say p, essentially partitioning s into two segments, say s_1, s_2. Then, we create two LineStrings t_1, t_2. The LineString t_1 is randomly chosen to be equal to either s_1 or s_2. Without loss of generality, let's assume that s_1 is chosen. The LineString t_2 is a LineString that is disjoint from s. The final target LineString t is defined by adjoining t_1 with t_2.

Gold Standard For the Spatial benchmark, the gold standard, could not be produced during the generation of the target dataset. As mentioned earlier, such an approach might lead to the creation of certain t_i that inadvertently have some relation with some s_j $(i \neq j)$, thus, the gold standard would be neither complete nor correct. To support completeness and correctness of the gold standard, the benchmark should check each generated target LineString t_i against all source LineStrings s_i for all possible relations that essentially amounts to implementing a system for the computation of the topological relations.

To avoid this problem, after all the source and target datasets are generated, we compute the gold standard using RADON [4]. RADON is a novel approach for rapid discovery of topological relations between geo-spatial resources. RADON combines space tiling, minimum bounding box approximation and a sparse index to achieve high scalability. RADON was evaluated 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 orders of magnitude. Listing 1.2 shows a small example of source, target datasets and the gold standard.

```

---------- SOURCE ----------

t1 a Trace .
t1 hasGeometry "LINESTRING (8.79232 48.92266, 8.79186 48.92246,
8.79135 48.92229, 8.79079 48.92219, 8.79012 48.92216, 8.78957
    48.92207,
8.78903 48.92185, 8.78848 48.92162, 8.78807 48.92133, 8.78782
    48.92097,
8.78771 48.92068, 8.78737 48.92031, 8.78695 48.92004)"
^^<http://strdf.di.uoa.gr/ontology#WKT> .

---------- TARGET ----------

t1' a Trace .
t1' hasGeometry "LINESTRING (8.78903 48.92185, 8.78848 48.92162,
8.78807 48.92133, 8.78782 48.92097, 8.78771 48.92068, 8.78737
    48.92031)"
^^<http://strdf.di.uoa.gr/ontology#WKT> .

---------- GOLD STANDARD (e.g. for COVERS) ----------

t1  t1'.
```

Listing 1.2: Spatial Benchmark Example

Key Performance Indicators (KPIs) The performance metric(s) in a benchmark determine the effectiveness and efficiency of the systems and tools. In this benchmark, we focus on the quality of the output in terms of standard metrics such as precision, recall and f-measure [6]. We also aim to quantify the performance (in ms) of the systems measuring the time needed to return all the results.

4 Experimental Results

In order to show the rate between the time performance and the population (10, 100, 1K and 10K) of the traces, but also the time difference between the topological relations, we executed a set of experiments for all the DE-9IM relations using the HOBBIT Platform[11]. Thus, we provide a comparative analysis with benchmarks produced using the *Spatial Benchmark* generator to assess and identify the capabilities of RADON and SILK, two of the state of the art systems. We should mention here that we are only presenting the time performance and not precision, recall and f-measure as all were equal to 1.0. Moreover, because of a problem encountered for SILK[12], we had to select traces no larger than 64 KB from all datasets in order to get a fair comparison for the systems under test.

Table 1 and Figures 5, 6 show the results of the executed experiments. Time is measured in milliseconds but for a better presentation of the results we use log(time). Relation *Disjoint*, needs the shortest time for the systems to return the results if the number of traces is small. However, as the number of traces increases, Silk seems to have difficulties linking all the traces fast enough, and the experiment stops due to the platform time limit (75 minutes)[13].

Intersects, *Touches* and *Overlaps*, seem to be the hardest relations for the systems as they need more time than *Equals*, *Crosses*, *Covers/Covered By* and *Contains/Within*, which need approximately the same time. Specifically, for *Touches* and *Intersects* we were not able to run experiments for the 10K traces source dataset (the target dataset consists of the same number of traces) as in the other cases the experiments stopped for both RADON and Silk. The reason that those relations need much more time is that our data contain LineStrings that have a large number of points in a rather small area. *Covers/Covered By* is not supported by Silk, but *Contains/Within* are the same when the datasets consist of LineStrings, thus, this was not an issue for the system. Generally, RADON seems to handle the growth of the dataset size smoother than Silk, meaning that when it comes to a small number of traces, RADON needs approximately the half time than Silk but as the size increases, RADON outperforms Silk by more than one order of magnitude.

[11] `http://master.project-hobbit.eu`

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

[13] The platform has a time limit for each benchmark to prevent it from binding the platform continuously.

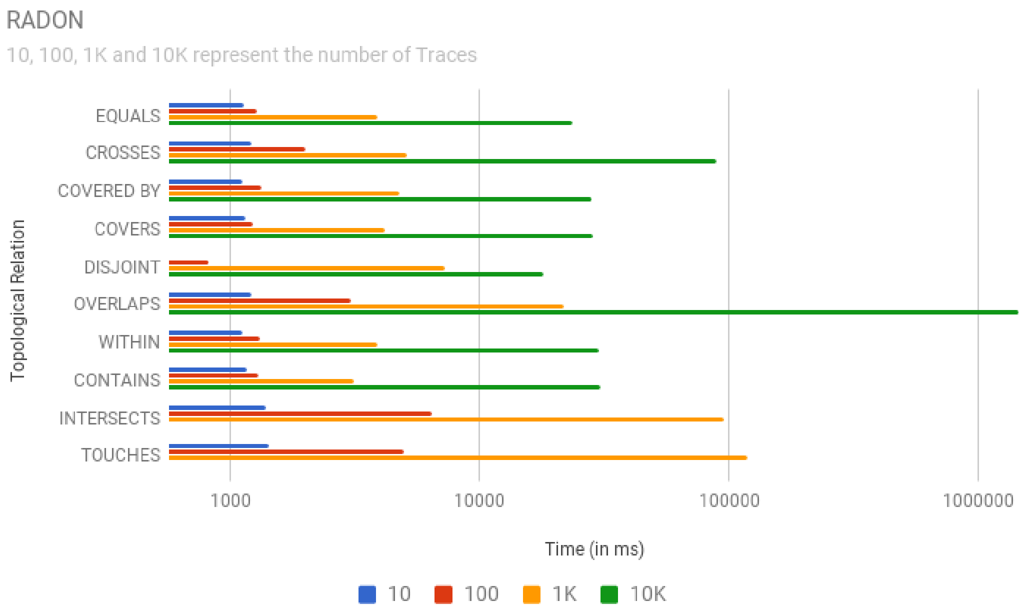


Fig. 5: RADON

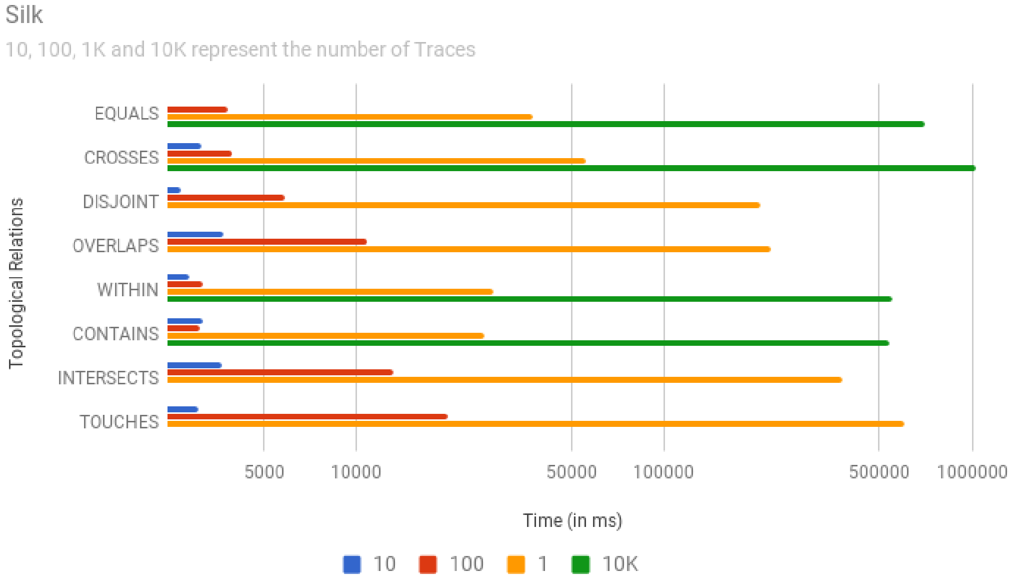


Fig. 6: Silk

RELATION	System	# 10	# 100	# 1K	# 10K
EQUALS	RADON	1139	1290	3891	23509
	Silk	2424	3804	37585	704821
CROSSES	RADON	1209	1996	5139	88799
	Silk	3126	3927	55691	1035194
COVERED BY	RADON	1124	1334	4793	28095
	Silk		Not Supported		
COVERS	RADON	1147	1235	4158	28673
	Silk		Not Supported		
DISJOINT	RADON	567	820	7269	18031
	Silk	2686	5843	206313	Platform time limit
OVERLAPS	RADON	1209	3047	21829	1452969
	Silk	3693	10790	221734	Platform time limit
WITHIN	RADON	1116	1311	3899	30233
	Silk	2859	3151	27933	550819
CONTAINS	RADON	1163	1294	3155	30560
	Silk	3170	3079	26094	538441
INTERSECTS	RADON	1400	6396	94733	Platform time limit
	Silk	3628	13175	381310	Platform time limit
TOUCHES	RADON	1436	4958	118104	Platform time limit
	Silk	3072	19786	601024	Platform time limit

Table 1: Time rate (in ms) while increasing population for each topological relation for RADON and Silk.

5 Conclusions

We presented HOBBIT's *Spatial benchmark*, a benchmark that checks whether the systems can identify DE-9IM (Dimensionally Extended nine-Intersection Model) topological relations. To the best of our knowledge, such benchmarks do not exist while the number of link discovery systems that identify links for spatial datasets are limited.

We evaluated two of the state of the art systems, RADON and Silk using the *Spatial Benchmark* generator to assess and identify their capabilities. Our results show that performing link discovery for some of the DE-9IM relations (touches, intersects, overlaps) takes prohibitive time. On the other hand, for the rest of the relations, the systems seem to need much less time. In future work, we aim to extend the implementation of the JTS extension in order to generate different classes of object geometries in addition to LineString. In addition, we will explore the possibility to include new evaluation metrics in the benchmark.

References

1. Axel-Cyrille Ngonga Ngomo. On link discovery using a hybrid approach. *Journal on Data Semantics*, 1(4):203–217, 2012.
2. T. Saveta, E. Daskalaki, G. Flouris, I Fundulaki, M. Herschel, and A.-C. Ngonga Ngomo. Pushing the limits of instance matching systems: A semantics-aware benchmark for linked data. In *WWW*, pages 105–106. ACM, 2015. Poster.

3. Christian Strobl. *Encyclopedia of GIS*, chapter Dimensionally Extended Nine-Intersection Model (DE-9IM), pages 240–245. Springer, 2008.
4. Mohamed Ahmed Sherif, Kevin Dreßler, Panayiotis Smeros, and Axel-Cyrille Ngonga Ngomo. RADON - Rapid Discovery of Topological Relations. In *Proceedings of The Thirty-First AAAI Conference on Artificial Intelligence (AAAI-17)*, 2017.
5. Panayiotis Smeros and Manolis Koubarakis. Discovering spatial and temporal links among rdf data. In *LDOW@ WWW*, 2016.
6. Cyril Goutte and Eric Gaussier. A probabilistic interpretation of precision, recall and f-score, with implication for evaluation. In *European Conference on Information Retrieval*, pages 345–359. Springer, 2005.