

# Scalable Spatial Topology Joins

Thanasis Georgiadis

Univ. of Ioannina & Archimedes, Athena RC  
Greece  
ageorgiadis@cs.uoi.gr

Nikos Mamoulis

Univ. of Ioannina & Archimedes, Athena RC  
Greece  
nikos@cs.uoi.gr

## Abstract

Identifying topological relations between complex spatial objects, such as polygons, is a fundamental problem with applications in geo-spatial interlinking, spatial databases, image retrieval, and more. The standard approach involves two steps: a filter step that checks for overlapping minimum bounding rectangles (MBRs) and a refinement step that computes the DE-9IM matrix to determine the precise relationship between objects. We propose an intermediate step that leverages precomputed raster representations of objects to reduce the need for costly DE-9IM matrix computations. Experimental results show that our method improves throughput by an order of magnitude for object pairs passing the MBR filter. More importantly, our approach scales efficiently, with its effectiveness increasing as object complexity grows.

## Keywords

spatial joins, topological relations, geospatial interlinking, raster approximations

## 1 Introduction

Topological relations between geometrical objects, illustrated in Figure 1(a) capture semantics that are unaffected by transformations of the data space, such as translation, rotation, and scaling. In GIS [9], they can be used in urban and transportation planning [18]. In *environmental studies*, detecting topological relations on climate, sustainability, and energy data (such as data offered by the European Environment Agency (EEA) [12]), can help understanding factors contributing to urban and global heating, pollution, and biodiversity decline. In *spatial databases* [22], topological relations are often used as predicates in selection queries and spatial joins [6]; besides, they have also been used in spatial query optimization [19]. In *geo-spatial interlinking* [25, 31], they are used to enrich and integrate knowledge graph databases with links between spatial entities.

In *image and multimedia databases*, relations between detected objects are valuable features, e.g., in medical image analysis [32]. Besides, content-based image retrieval based on topological queries [26, 36] finds interesting arrangements of objects in images. Proteins in large biological databases are also topologically related [16].

The Dimensionally Extended 9-intersection model (DE-9IM) [7, 37] is the de-facto standard for capturing the possible spatial relations between shapes. DE-9IM has been widely used in GIS (ArcGIS [11], QGIS [1]) and spatial DBMS (PostGIS [28], Oracle Spatial [24]), while it has been implemented in popular geometry libraries (JTS [21], GEOS [15], boost [3]). A typical approach to detect whether two arbitrary spatial objects (e.g., polygons) intersect, or their topological relation in general, is to apply a cheap *filter step* using the *minimum bounding rectangles* (MBRs)

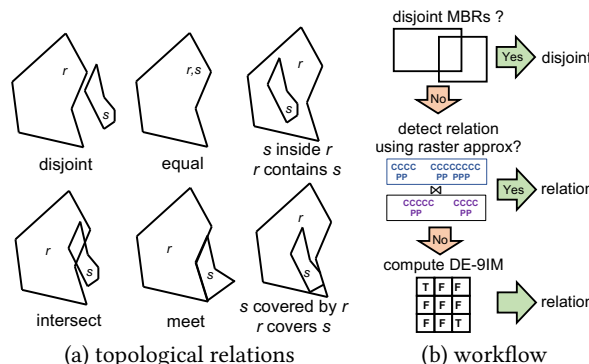


Figure 1: Topological relations and proposed workflow

of the objects [6, 31]. If the MBRs do not intersect, then the two objects are definitely disjoint. Otherwise, computing the DE-9IM matrix for the object geometries is necessary to determine their precise topological relation. This requires  $O(n \log(n))$  time (the same as the complexity of polygon-polygon intersection test), where  $n$  is the number of vertices of the two shapes [8].

**Contribution** We propose an intermediate step in the pipeline of topology detection, after the MBR filter step, which takes advantage of precomputed spatial object approximations as lists of raster intervals [14]. We define a set of *relations* (e.g., overlap) between lists of raster intervals, which can be evaluated in linear time by merge-joining the lists. Then, depending on how the MBRs of two objects intersect, we propose specialized filters, which verify a sequence of list relations to potentially confirm the topological relation between the objects, without having to access and process their exact geometries. Hence, compared to previous work which uses raster object approximations [13, 14, 42] to detect spatial *intersection* only, we exploit such approximations in full for the detection of the most specific topological relation between two objects. When applied on benchmarking datasets (Tiger, OSM), our approach boosts the overall throughput of spatial topology joins up to one order of magnitude compared to state-of-the-art geo-spatial interlinking methods [25, 31] and up to several times compared to using raster approximations for intersection detection [14].

The workflow of our approach is illustrated in Figure 1(b). For each pair of objects whose MBRs intersect (i.e., pairs produced by an algorithm handling the filter step [27, 39]), we perform merge-join operations on their APRIL approximations, modeled as interval lists, to potentially determine their topological relationship. If the raster approximations are insufficient, we compute the DE-9IM matrix as a fallback. A key advantage of our method is its scalability—its effectiveness improves with increasing object complexity, enabling the detection of hundreds of thousands of topological relations per second for complex object pairs. Additionally, for most comparisons, our approach avoids loading full object geometries, significantly reducing data access costs.

EDBT '26, Tampere (Finland)

© 2025 Copyright held by the owner/author(s). Published on OpenProceedings.org under ISBN 978-3-98318-102-5, series ISSN 2367-2005. Distribution of this paper is permitted under the terms of the Creative Commons license CC-by-nc-nd 4.0.

## 2 Background and Related Work

### 2.1 DE-9IM

DE-9IM [7] breaks down each of the input geometries  $r$  and  $s$  into three parts: the *interior*, the *boundary* and the *exterior*. The intersection between each pair of parts is computed to fill in a  $3 \times 3$  matrix, where each row corresponds to a part of  $r$  and each column to a part of  $s$ . Each element of the matrix is a boolean value (T, F) that indicates intersection between the corresponding parts. A DE-9IM matrix can be flattened to a 9-element string code, where the first 3 values are the first row, the next 3 values the second row, etc. For example, the string code for the two objects  $r$  and  $s$  in the disjoint relation shown in Figure 1(a) is FFFFTTTT. To compute the DE-9IM matrix, plane sweep [29] or overlay of trapezoidal decompositions [8] can be used.

As shown in Table 1, each topological relation can be detected by applying one or more DE-9IM masks, i.e., strings having T, F, or \* at each position, where \* denotes any of {T, F}. If the DE-9IM string code matches a mask, then the object pair satisfies the corresponding topological relation. Figure 2 shows a Venn diagram of the 8 topological relations and their relationships. The generalization hierarchy of relations is reflected by the relationships between their masks. For example, the *covers* masks are all included in some the *intersects* masks.

Table 1: DE-9IM masks of topological relations.

	DE-9IM mask			
<b>disjoint</b>	FF*FF****			
<b>intersects</b>	T*****	*T*****	***T*****	****T****
<b>covers</b>	T*****FF*	*T****FF*	***T**FF*	****T**FF*
<b>covered by</b>	T*F**F***	*TF**F***	**FT*F***	**F*TF***
<b>equals</b>	T*F**FFF*			
<b>contains</b>	T*****FF*			
<b>inside</b>	T*F**F***			
<b>meets</b>	FT*****	F**T*****	F***T*****	

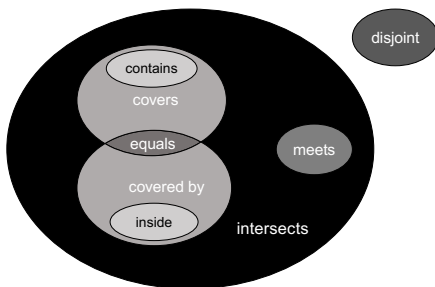


Figure 2: Venn diagram of the 8 topological relations.

### 2.2 MBR Filtering

The detection of spatial relations between objects (or the evaluation of spatial joins) is typically conducted in two steps. At the *filter* step, the MBRs of the objects are compared in  $O(1)$  time, to possibly filter the pair from further consideration. As MBRs are conservative approximations [6], they can only be used to detect disjointness. When a massive number of relations need to be detected, as in geo-spatial interlinking [31], the filter step can be evaluated efficiently as a spatial join between MBRs [39]. This problem has been well-studied with scalable solutions to the number of objects for in-memory data [38, 39] and disk-based

[5, 27] or distributed data [10, 40]. Still, for the verification of the exact relation between two objects whose MBRs intersect, the current practice is to compute their DE-9IM matrix. In view of the high cost of DE-9IM matrix computations, Papadakis et al. [25] suggested to examine the intersecting MBRs pairs in an order that would maximize the chances of detecting non-disjoint relations. This work is orthogonal to our objective of reducing the computational complexity of detecting topological relations. Learning techniques have also been developed to improve the efficiency of link completion in spatial knowledge graphs [41], some through link prediction [20, 23, 30]. These methods are approximate while having a high training cost.

### 2.3 Intermediate Filters for Spatial Intersection

In spatial intersection joins [5], to defer the refinement step, previous work [6, 13, 14, 42] suggests the pre-computation and use of (finer than MBR) object approximations in an *intermediate* filter, applied after the MBR filter, to determine whether (a) the objects are definitely disjoint, (b) the objects definitely intersect, or (c) we cannot decide and we need to apply a refinement step.

The approximations in [6] are simple shapes (e.g., convex polygons) that cover the complex object. In [42] raster object approximations have been suggested. The Raster Intervals (RI) [13] technique builds upon [42] to define a fine-grained global grid over the data space, whose cells are enumerated by a Hilbert curve [17]. For each object, consecutive cells that overlap it are modeled as intervals. Rasters have also been used for spatial statistics [33–35].

We adopt APRIL [14], a space-economic, fast-to-construct and fast-to-process approximation. Consider a fine-grained global grid with Hilbert-enumerated cells. For each object  $o$ , we define *two* sorted interval lists. The *Progressive* ( $P$ ) list contains intervals that include only the cells entirely inside  $o$  (i.e. full cells); The cells in  $P$  always cover less space in the grid than  $o$ , thus  $P$  is a *progressive approximation* [6] of  $o$ . The *Conservative* ( $C$ ) list of  $o$  has the intervals that include all cells partially or fully covered by  $o$ . Figure 3 shows the  $P$  and  $C$  interval lists of two example polygons. The  $P$  and  $C$  interval lists of each object are generated efficiently in a data pre-processing step (conducted once per object).

As shown in [14], for two objects  $r$  and  $s$ , merge-joining their  $P$  ( $r_P, s_P$ ) and  $C$  ( $r_C, s_C$ ) lists can help to identify whether the two objects intersect. If there is no pair of intervals ( $i_C \in r_C, j_C \in s_C$ ) such that  $i_C$  and  $j_C$  overlap, as in the example of Fig. 3,  $r$  and  $s$  are definitely disjoint. Otherwise, we join  $r_C$  with  $s_P$  to potentially find a pair ( $i_C \in r_C, j_P \in s_P$ ); if such a pair exists, then  $r$  definitely intersects  $s$ ; if not, we repeat with  $r_P$  and  $s_C$ . Failing all tests renders the object pair *inconclusive*, to be handled at a refinement stage.

### 3 Fast detection of topological relations

This section describes our methodology for efficiently determining topological relations. Formally, given two objects  $r$  and  $s$  whose MBRs intersect (i.e.  $r$  and  $s$  pass the filter step of spatial join), the *find relation* problem aims to identify their most specific topological relationship. We assume that the  $P$  and  $C$  approximations [14] of  $r$  and  $s$  have been precomputed. Depending on how the MBRs of  $r$  and  $s$  intersect, we apply a tailored sequence of binary merge-join operations on  $P$  and  $C$  interval lists to solve the *find relation* problem without having to compute the DE-9IM matrix.

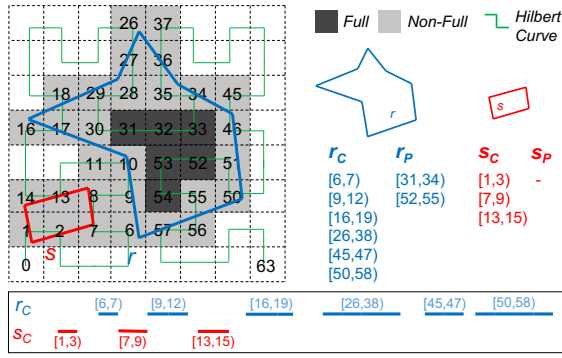


Figure 3: Progressive and Conservative interval lists.

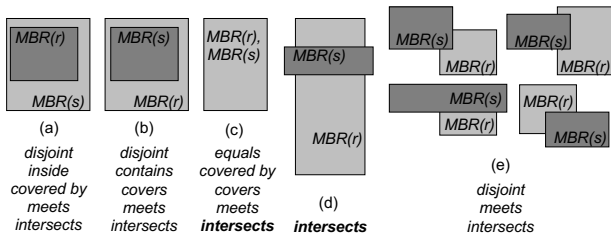


Figure 4: The candidate and definite (bold) topological relations of two polygons, based on how their MBRs intersect.

### 3.1 MBR Filter

The possible topological relations between a pair of shapes,  $r$  and  $s$ , can be constrained based on how their MBRs intersect. For example, consider Figure 4(a), where the MBR of  $r$  is fully contained within the MBR of  $s$ . From this, we can infer that  $r$  and  $s$  cannot be equal, because if they were equal, their MBRs would also be equal. Additionally,  $r$  cannot contain or cover  $s$ , eliminating 3 out of the 8 possible relations. Figure 4 illustrates, for each type of MBR intersection, the possible topological relations between the corresponding objects. Relations marked in bold font are certain to hold. A notable case is Figure 4(d), where the only valid relation is *intersects*, and no refinement step is necessary to specialize it.

### 3.2 Intermediate Filter

Our enhanced MBR filter (Sec. 3.1) allows for more specialized handling in the intermediate filter and refinement. A pair of objects with intersecting MBRs is forwarded to the appropriate intermediate filter that focuses only on the possible relations, minimizing the computations that need to be performed. The intermediate filters perform merge-join operations on the objects' sorted  $C$ - and/or  $P$ -lists based on the candidate relations. We present the workflows of the intermediate filters using the following relations between two *interval* lists  $X$  and  $Y$  taken from  $\{r_P, r_C, s_P, s_C\}$ .

' **$X, Y$  overlap**' At least one pair of intervals  $x \in X, y \in Y$  intersect; i.e.  $x$  and  $y$  include at least one common cell identifier.

' **$X, Y$  match**' The two interval lists  $X$  and  $Y$  are identical; i.e., for each  $x \in X$  there is a  $y \in Y$ , such that  $x = y$  and vice versa.

' **$X$  inside  $Y$** ' Every interval in  $X$  is contained in one interval of  $Y$ ; i.e., for each  $x = [x_{start}, x_{end}] \in X$ , there exists an interval  $y = [y_{start}, y_{end}] \in Y$ , such that  $x_{start} \geq y_{start} \wedge x_{end} \leq y_{end}$ .

' **$X$  contains  $Y$** ' is defined inversely to ' **$X$  inside  $Y$** '; every interval in  $Y$  is contained in one interval of  $X$ .

Note that each of the above relations takes linear time to evaluate, as the intervals within each list are disjoint. Moreover, the number of intervals per list is expected to be low (in the order of square root of the number of cells that intersect with the object) [14].

#### Algorithm 1 Evaluation of *find relation*.

```

Require:  $r, s$ 
1: function MBRFILTER( $r, s$ )
2:   if  $MBR(r), MBR(s)$  are disjoint then
3:     return disjoint
4:   else if  $MBR(r) = MBR(s)$  then
5:     result  $\leftarrow$  IFEQUALS( $r, s$ )
6:     if result = coveredby, covers then
7:       return result
8:     else
9:       return Refine(covers, coveredby, intersects)
10:    end if
11:  else if  $MBR(r)$  inside  $MBR(s)$  then
12:    result  $\leftarrow$  IFINSIDE( $r, s$ )
13:    if result = disjoint, inside, intersects then
14:      return result
15:    else if result = refinside then
16:      return Refine(inside, coveredby, intersects)
17:    else
18:      return Refine(disjoint, inside, coveredby, meets, intersects)
19:    end if
20:  else if  $MBR(r)$  contains  $MBR(s)$  then
21:    result  $\leftarrow$  IFCONTAINS( $r, s$ )
22:    if result = disjoint, contains, intersects then
23:      return result
24:    else if result = refcontains then
25:      return Refine(contains, covers, intersects)
26:    else
27:      return Refine(disjoint, contains, covers, meets, intersects)
28:    end if
29:  else if  $MBR(r), MBR(s)$  cross then
30:    return intersects
31:  else
32:    result  $\leftarrow$  IFINTERSECT( $r, s$ )
33:    if result = disjoint, intersects then
34:      return result
35:    else
36:      return Refine(disjoint, meets, intersects)
37:    end if
38:  end if
39: end function

```

Algorithm 1 describes our proposed approach for solving *find relation* problems using MBRs and  $C/P$  raster approximations. The algorithm only handles candidates whose MBRs intersect (line 2). Depending on how the MBRs of  $r$  and  $s$  intersect, one of the *individual intermediate filters* (IF), shown as flow diagrams in Figure 5 is applied. Each IF returns either the most specific relation, in which case a *refinement step* that would compute the DE-9IM matrix is unnecessary, or a set of possible relations. In the latter case, the DE-9IM matrix is computed and then compared against the masks of possible relations in a specific-to-general order, to find the most specific topological relation between  $r$  and  $s$ . We now elaborate on each MBR intersection case.

**Equal MBRs** If the MBRs are equal (Figure 4(c)), then the pair is forwarded to the *IFEquals* intermediate filter, shown first in Figure 5. This filter is able to detect exactly, i.e., without refinement, the *covers* and *covered by* relations. If the  $C$  lists of the two objects are identical, then the pair is forwarded to refinement for the most specific of *equals, covered by, covers, and intersects*. Although the objects definitely *intersect*, this might not be their most specific relation. If the  $C$  lists do not match, the algorithm proceeds to check all possible relations. In general, if the most specific relation cannot be detected from the  $C$  and  $P$  lists of the two objects, the pair is forwarded to *selective refinement*, where the not definite relations are verified after the DE-9IM matrix computation.

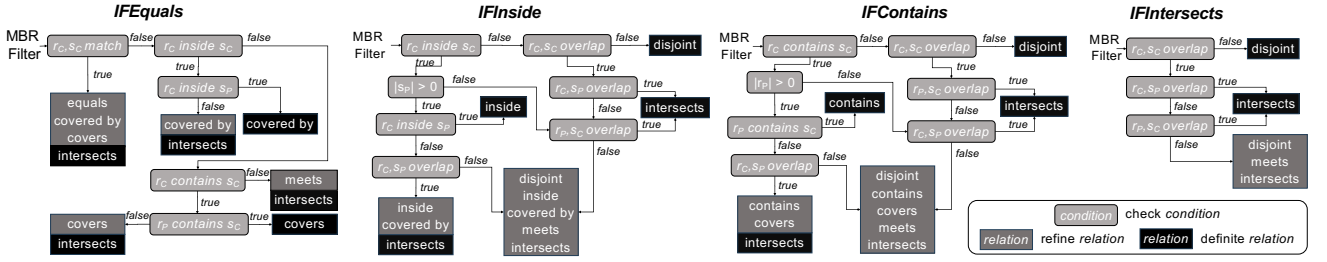


Figure 5: The intermediate filters for the different cases of intersections between the MBRs of  $r$  and  $s$ .

**One MBR inside the other** When one of the MBRs is contained inside the other (Figure 4(a) or 4(b)), Algorithm 1 forwards the candidate pair to *IFInside* or *IFContains* intermediate filters, respectively, based on which MBR is inside the other. The two filters work in the same way, with the difference being that the first one looks for  $r$  inside  $s$  or  $r$  covered by  $s$ , whereas the second looks for  $r$  contains  $s$  or  $r$  covers  $s$ , besides *disjoint*, *meets*, and *intersects*. The check to determine whether an interval list  $r_C$  is completely contained in a list  $s_P$  has  $O(|r_C| + |s_P|)$  time complexity.

**Other cases** If the two MBRs cross each other, as in Figure 4(d), we definitely know that the most specific relation is *intersects* and no intermediate filter or refinement is necessary. All other cases, shown in Figure 4(e) are handled by the *IFIntersect* intermediate filter, which detects *disjoint*, *meets* and *intersects* relations.

**Complexity of Algorithm 1** Up to four relations between the  $P$  and  $C$  interval lists of  $r$  and  $s$  are evaluated in each of the flow diagrams of Figure 4, until we can conclude about the topological relation(s) between  $r$  and  $s$ . For each relation, the cost is linear to the lengths of the two merge-joined lists, because the intervals in a list are disjoint to each other. Hence, the overall cost of the IF applied on a given pair of objects  $r$  and  $s$  is  $O(|r_P| + |r_C| + |s_P| + |s_C|)$ .

### 3.3 Relate

In certain scenarios, instead of looking for the most specific topological relation between two objects, we need to quickly check if a given relation is satisfied. For example, spatial joins may take a topological relation as a predicate. Formally, given a pair  $(r, s)$  of objects such that  $MBR(r)$  intersects  $MBR(s)$  and a topological relation  $p$  (e.g.,  $p=meets$ ), the  $relate_p$  problem finds whether  $r$  and  $s$  satisfy  $p$ . Figure 6 (left) shows the intermediate filter's flow diagrams for  $relate$  predicates that are specializations of *intersects*. The figure is self-explanatory; for a given predicate a sequence of merge-join operations are applied on the  $C$  and/or  $P$  lists to potentially verify if the corresponding relation definitely holds or not. For instance, for a pair  $(r, s)$  of objects, if not  $r_C$  inside  $s_C$ , then definitely  $r$  is not inside  $s$ ; if  $r_C$  inside  $s_P$ , then definitely  $r$  is inside  $s$ . In all other cases,  $(r, s)$  is sent to the refinement step.

## 4 Experimental Analysis

We evaluated the performance of our topological relation detection approach, compared to the current practice and the use of intermediate filters for intersection detection [14]. We compare the following approaches:

**ST2: standard 2-phase** This method applies the current MBR filter+refinement practice used in most previous work that detect topological relations [25, 31]. If the object MBRs do not intersect,

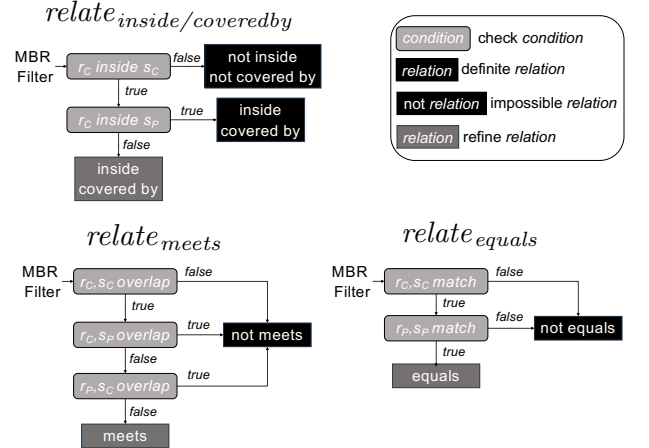


Figure 6: Intermediate filter flow diagrams for  $relate_p$  tests.

Table 2: Description of datasets.

Dataset	Entity type	# polygons	Size (MB)	MBRs (MB)	$P + C$ (MB)
TL	US Landmarks	123K	52.5	3.7	6.3
TW	US Water areas	2.25M	1.2K	68.6	73.0
TC	US Counties	3.04K	112.8	0.1	15.4
TZ	US Zip Codes	26.1K	587	0.8	170.1
OBE	EU Buildings	90.4M	10.9K	2.8K	2.3K
OLE	EU Lakes	1.96M	1.1K	59.8	82.0
OPE	EU Parks	7.17M	3.7K	218.8	389.0
OBN	NA Buildings	9.38M	1.3K	286.3	201.0
OLN	NA Lakes	4.02M	2.5K	122.7	133.0
OPN	NA Parks	999K	767.4	30.5	60.0

then the pair is disjoint, otherwise we compute the DE-9IM matrix to determine the topological relation.

**OP2: optimized 2-phase** This method uses the relation between the MBRs, as described in Sec. 3.1 to limit the possible topological relations and, in turn, reduce the number of DE-9IM masks to compare with the DE-9IM matrix.

**APRIL: optimized MBR filter + APRIL intermediate filter + refinement** This method applies APRIL [14] to detect non-intersection between two objects whose MBRs intersect, before applying the refinement step. As the intermediate filter in this case cannot detect more special relations than *intersects*, the DE-9IM matrix computation is necessary even for pairs that are found to definitely intersect, to detect a potentially more specific relation.

**P+C: optimized MBR filter + Progressive/Conservative filter + refinement** This is our algorithm, presented in Section 3.

**Table 3: Semantically meaningful dataset combinations for the *find relation* and *relate<sub>p</sub>* problems.**

Datasets	TL-TW	TL-TC	TC-TZ	OLE-OPE	OLN-OPN	OBE-OPE	OBN-OPN
Candidate pairs	63.3K	168K	65.7K	5.18M	2.77M	79.3M	2.17M

#### 4.1 Datasets & Setup

We used popular benchmarking datasets from the TIGER 2015 and Open Street Map (OSM) collections [10], containing real-world areas from the USA and the entire globe, respectively. As the *find relation* problem makes sense for objects that exist in the same region, we also split each OSM dataset to two parts, one for Europe and one for North America (the continents with the highest concentration of objects). Additionally, we cropped the TIGER datasets to include only entities in the contiguous (lower 48) United States. Table 2 describes the datasets that we used, and the space occupied by their polygons and their approximations. T- and O- prefixes of datasets denote TIGER and OSM collections, respectively.

Table 3 shows dataset combinations that we used in our experiments. For each combination, the table also shows the number of object pairs that pass the MBR filter. The dataset pairs represent real-world scenarios where the detection of topological relations between areas is meaningful:

- TL-TW: US landmarks (TL) may include parks, lakes, canyons, buildings, etc., so geospatially interlinking pairs from TL and TW may reveal unexpected relations and statistics about all landmark and water areas in the US.
- TL-TC: As US counties (TC) are relatively large areas, it is expected that most of the relations in this scenario will be *inside*. However, unique cases can be found, such as cross-county landmarks or even landmarks containing entire counties.
- TC-TZ: This pair provides insights about the county and zip code relations in the US.
- OBx-OPx(E/N): Topological relations in this scenario may provide insight on the amount of human intervention (buildings) in green areas (parks) in Europe and North America.
- OLx-OPx(E/N): Interlinking entities between these datasets provides information about the presence of water elements (currents, lakes, marshes etc.) in the parks of Europe and North America.

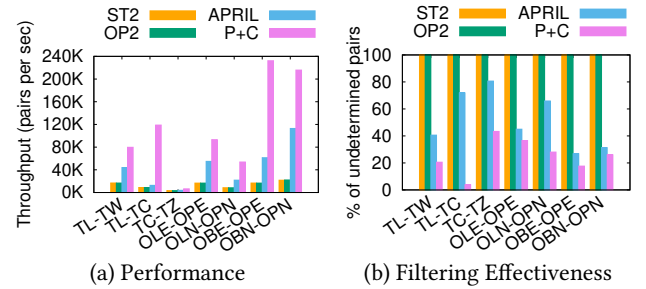
For each of the scenarios above, we run a spatial intersection join algorithm [39] that produces the pairs of objects whose MBRs intersect. Each such pair was fed to our topological relation detection pipeline. For each of the evaluated methods, we measured their *throughput*; i.e., the number of MBR-filtered pairs that the method processed per second. We did not account for the cost of the filter step (i.e., the time to produce the pairs of MBRs that intersect), which is negligible compared to the cost of identifying the topological relation for each such pair.

All raster approximations that we used for the experiments, were created using independent  $2^{16} \times 2^{16}$  grids overlaid on each data scenario’s dataspace. As shown in Table 2, the *Progressive* and *Conservative* interval lists occupy much less space compared to the polygons. For the refinement, we use the boost geometry’s *relation* function [4] that calculates the DE-9IM matrix for two input geometries (boost’s implementation was found to be much faster than GEOS [15]). We ran our experiments on a machine

with a 3.6GHz Intel i9-10850k and 64GB of RAM, running Linux. The code was written in C++ and compiled with the -O3 flag.

#### 4.2 Performance

Figure 7(a) shows the throughput of the compared methods on the test dataset pairs. Observe that, in practice, there is no improvement when using an optimized MBR filter (OP2), compared to directly moving the pair to refinement (ST2). This is because the computation of the DE-9IM matrix is the bottleneck of the entire process, so the savings of OP2 by reducing the number of masks (relations) that are being checked are marginal. Using the APRIL intermediate filter greatly improves the throughput, by detecting many cases of object pairs that are disjoint. Our P+C method (Sec. 3) guides the comparison between the *C* and *P* lists of the objects, and detects more topological relations than APRIL (on top of intersects [14]). Overall, our P+C approach improves the throughput of *find relation* by one order of magnitude compared to the current practice (ST2, OP2) [25, 31] and by a few to several times compared to using the APRIL intermediate filter [14].

**Figure 7: Throughput and Effectiveness for *find relation*.**

The improvement in the throughput when transitioning from 2 stages in the pipeline (ST2, OP2) to 3 (APRIL, P+C) is related to the effectiveness of the intermediate filters. For each pair of datasets, we measure filter effectiveness in terms of undetermined pairs, i.e. pairs that need to be refined using DE-9IM to detect their topological relation. As shown in Figure 7(b), the introduction of the APRIL intermediate filter reduces the DE-9IM matrix computations to around half compared to ST2, OP2, on average. Our specialized intermediate filter workflows (Sec. 3) reduce the object pairs to be refined even more, to about 25% on average.

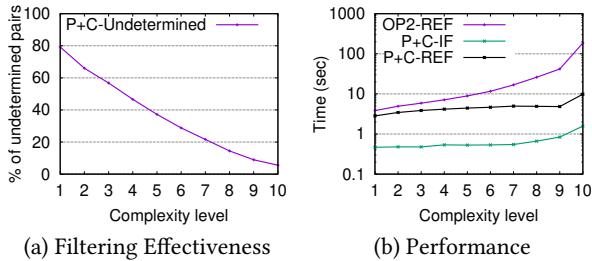
#### 4.3 Scalability

We tested the scalability of our approach (P+C) with respect to the complexities of the object pairs whose topological relation is being detected. For a pair of polygons *r* and *s*, we define complexity as the sum of their vertices, as this determines the cost of computing intersections between their parts [8], and in turn, the cost of DE-9IM matrix computations. We selected a fairly large scenario (OLE-OPE) and divided its post-MBR candidate pairs into 10 groups of increasing complexity, shown in Table 4. The ranges were selected, such that each complexity level contains around the same number of object-pairs; hence, we test the scalability of our method on similar data workloads per complexity level.

Figure 8(a) shows P+C’s intermediate filter effectiveness in detecting relations without DE-9IM matrix computations for polygon pairs from the OLE-OPE scenario, having different complexities. High complexity pairs are resolved more frequently

**Table 4: OLE-OPE post-MBR filter polygon pairs, grouped by complexity level (sum of their vertex count).**

Complexity level	Sum of vertices	Pair count
1	[8,41]	525K
2	[42,67]	518K
3	[68,104]	513K
4	[105,163]	520K
5	[164,265]	520K
6	[266,447]	517K
7	[448,786]	518K
8	[787,1354]	518K
9	[1355,2629]	518K
10	[2630,60398]	518K

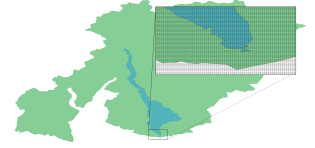
**Figure 8: Find relation filter effectiveness and total cost for polygons pairs grouped by complexity level.**

by the intermediate filters of P+C compared to low complexity pairs. This is due to the fact that most of the intermediate filters of P+C utilize the polygons'  $P$  lists to detect early definite topological relations. Raster-based interval approximations of small polygons usually result in very few, if any at all, full cells and progressive intervals, rendering refinement necessary to detect their topological relation. As shown in Figure 8(a), 4 out of 5 pairs of complexity level 1 in the OLE-OPE scenario need to be geometrically refined, contrary to pairs of complexity level 10 where only around 5% are refined.

Polygon pairs of low complexity are cheap to refine and, as complexity increases, the refinement becomes more expensive, with polygons of complexity level 10 being the bottleneck of the entire pipeline. This is reflected in Figure 8(b), where the cost of OP2 which refines almost all pairs increases superlinearly with the complexity level. On the other hand, the cost of P+C is almost insensitive to the complexity of the pairs due to the fact that the increase in the refinement per pair is compensated by the decrease of the number of pairs that need to be refined, as seen in Figure 8(a). This experiment unveils an important value of our approach, which achieves its highest filtering power on objects that have the largest need for it. Besides saving computations, our approach also manages to load less data compared to OP2. For example, in the OLE-OPE scenario, the P+C approach accesses only 48.5% of the unique objects from OLE and OPE that are accessed by OP2, as the topological relations involving the remaining objects are detected by the P+C intermediate filters.

Figure 9 showcases a pair of objects at complexity level 10, where their relation (*inside*) is identified by the P+C intermediate filter and the DE-9IM matrix computation is avoided. This pair is forwarded to refinement by ST2, OP2, and APRIL, as these methods cannot detect the most specific relation (APRIL would only be able to detect intersection). As a result, our P+C approach is 50x faster than the other methods for this pair. The pair represents a lake residing inside a park; even though the park is a

	Lake	Park
Vertices	2240	2616
MBR area	0.0616	0.5030
C-intervals	498	1793
P-intervals	481	1845



(a) Statistics

(b) Illustration

**Figure 9: A level-10 complexity pair of a lake (blue) residing inside a park (green), from the OLE-OPE scenario.****Table 5: Throughput (pairs/sec) of find relation and relate<sub>p</sub> methods using our P+C approach (OLE-OPE).**

Method	Equals	Meets	Inside
find relation	93160.2	93160.2	93160.2
relate <sub>p</sub>	107265.6	7000989.2	565509.7

lot bigger than the lake, they both have an adequate amount of  $P$  intervals, due to the fine-grained grid ( $2^{16}$  cells per dimension) that was used to create their approximations.

#### 4.4 Relate Performance

In the last experiment, we compare the effectiveness of our  $relate_p$  algorithms for specific topological predicates  $p$  (Section 3.3) against our general  $find relation$  algorithm (Section 3.2). We chose the OLE-OPE scenario because of the two datasets' relative balance in terms of polygon size and complexity. For three different predicates  $p$ , Table 5 shows that the throughput for  $find relation$  is independent of  $p$ , as the algorithm does not consider  $p$ . On the other hand, the throughput of  $relate_p$  queries is sensitive to the predicate, as a different intermediate filter is used in each case. The more focused  $relate_p$  approach is faster in all tested cases, due to the specialized steps it takes, depending on  $p$ . The difference is huge for some predicates (e.g., *meets*), where non-satisfaction is fairly easy to identify using the object approximations.

## 5 Conclusions

This paper presents a scalable technique for detecting topological relations between complex spatial objects (i.e., polygons) by leveraging raster approximations to minimize expensive DE-9IM matrix computations. Our approach not only reduces the number of object pairs requiring refinement but also becomes increasingly effective as object complexity — and thus refinement cost — grows.

In the future, we plan to integrate our method into existing link discovery frameworks such as Silk [2] to enable faster user-defined spatial link generation, contributing to the expansion of geospatially linked data available online. Additionally, implementing our approach in open-source spatial DBMSs (e.g., PostGIS [28]) could enhance the performance of topological relation queries, and, in turn the usability of systems and applications that handle complex spatial data and, in turn, improve the usability of systems and applications that handle complex spatial data.

## Acknowledgements

Work supported by project MIS 5154714 of the National Recovery and Resilience Plan Greece 2.0 funded by the European Union under the NextGenerationEU Program.

## Artifacts

We provide full access to our open-source code on GitHub along with the datasets we used and scripts to reproduce the experiments of this paper. Detailed instructions can be found in the repository's README.

Link: <http://github.com/ThanGeo/topology-detection>

## References

- [1] 2024. QGIS. [www.qgis.org](http://www.qgis.org).
- [2] 2024. Silk. <http://www.silkframework.org/>.
- [3] Boost. 2025. Boost C++ Libraries. <http://www.boost.org/>.
- [4] Boost.Geometry Development Team. 2023. *Boost.Geometry*. Boost. [https://www.boost.org/doc/libs/1\\_84\\_0/libs/geometry/doc/html/index.html](https://www.boost.org/doc/libs/1_84_0/libs/geometry/doc/html/index.html)
- [5] Thomas Brinkhoff, Hans-Peter Kriegel, and Ralf Schneider. 1993. Comparison of Approximations of Complex Objects Used for Approximation-based Query Processing in Spatial Database Systems. In *Proceedings of the Ninth International Conference on Data Engineering, April 19-23, 1993, Vienna, Austria*. IEEE Computer Society, 40–49.
- [6] Thomas Brinkhoff, Hans-Peter Kriegel, Ralf Schneider, and Bernhard Seeger. 1994. Multi-Step Processing of Spatial Joins. In *Proceedings of the 1994 ACM SIGMOD International Conference on Management of Data, Minneapolis, Minnesota, USA, May 24-27, 1994*. ACM Press, 197–208.
- [7] Eliseo Clementini, Paolino Di Felice, and Peter van Oosterom. 1993. A Small Set of Formal Topological Relationships Suitable for End-User Interaction. In *Advances in Spatial Databases, Third International Symposium, SSD'93, Singapore, June 23-25, 1993, Proceedings (Lecture Notes in Computer Science, Vol. 692)*. Springer, 277–295.
- [8] Mark de Berg, Otfried Cheong, Marc J. van Kreveld, and Mark H. Overmars. 2008. *Computational geometry: algorithms and applications, 3rd Edition*. Springer.
- [9] Michael N. DeMers. 2008. *Fundamentals of geographic information systems (4. ed.)*. Wiley.
- [10] Ahmed Eldawy and Mohamed F. Mokbel. 2015. SpatialHadoop: A MapReduce framework for spatial data. In *31st IEEE International Conference on Data Engineering, ICDE 2015, Seoul, South Korea, April 13-17, 2015*. IEEE Computer Society, 1352–1363.
- [11] ESRI. 2024. ArcGIS. [www.esri.com/en-us/arcgis/products/arcgis-pro](http://www.esri.com/en-us/arcgis/products/arcgis-pro).
- [12] European Environment Agency. [n. d.]. Data Hub. <https://www.eea.europa.eu/en/datahub>
- [13] Thanasis Georgiadis and Nikos Mamoulis. 2023. Raster Intervals: An Approximation Technique for Polygon Intersection Joins. In *Proceedings of the 2023 ACM SIGMOD International Conference on Management of Data, Seattle, Washington, USA, June*.
- [14] Thanasis Georgiadis, Eleni Tzirita Zacharatos, and Nikos Mamoulis. 2025. Raster interval object approximations for spatial intersection joins. *VLDB J.* 34, 1 (2025), 8.
- [15] GEOS contributors. 2021. *GEOS coordinate transformation software library*. Open Source Geospatial Foundation. <https://libgeos.org/>
- [16] Lin Guo, Jayavel Shanmugasundaram, and Golan Yona. 2007. Topology Search over Biological Databases. In *2007 IEEE 23rd International Conference on Data Engineering*. 556–565. doi:10.1109/ICDE.2007.367901
- [17] David Hilbert. 1891. Über die stetige Abbildung einer Linie auf ein Flächenstück. *Mathematische Annalen* 38, 1 (1891), 459–460.
- [18] Ick Hoi Kim, Chen-Chieh Feng, and Yi-Chen Wang. 2017. A simplified linear feature matching method using decision tree analysis, weighted linear directional mean, and topological relationships. *Int. J. Geogr. Inf. Sci.* 31, 5 (2017), 1042–1060.
- [19] Xuemin Lin, Qing Liu, Yidong Yuan, and Xiaofang Zhou. 2003. Multiscale Histograms: Summarizing Topological Relations in Large Spatial Datasets. In *Proceedings of 29th International Conference on Very Large Data Bases, VLDB 2003, Berlin, Germany, September 9-12, 2003*. Morgan Kaufmann, 814–825.
- [20] Ben Liu, Miao Peng, Wenjie Xu, Xu Jia, and Min Peng. 2024. UniLP: Unified Topology-aware Generative Framework for Link Prediction in Knowledge Graph. In *Proceedings of the ACM on Web Conference 2024, WWW 2024, Singapore, May 13-17, 2024*. ACM, 2170–2180.
- [21] Eclipse Foundation LocationTech group. 2024. JTS Topology Suite. [www.github.com/locationtech/jts](http://www.github.com/locationtech/jts).
- [22] Nikos Mamoulis. 2011. *Spatial Data Management*. Morgan & Claypool Publishers.
- [23] Genivika Mann, Alishiba Dsouza, Ran Yu, and Elena Demidova. 2023. Spatial Link Prediction with Spatial and Semantic Embeddings. In *The Semantic Web - ISWC 2023 - 22nd International Semantic Web Conference, Athens, Greece, November 6-10, 2023, Proceedings, Part I (Lecture Notes in Computer Science, Vol. 14265)*. Springer, 179–196.
- [24] Oracle. 2024. Oracle Spatial Database. [www.oracle.com/database/spatial/](http://www.oracle.com/database/spatial/).
- [25] George Papadakis, Georgios M. Mandilaras, Nikos Mamoulis, and Manolis Koubarakis. 2021. Progressive, Holistic Geospatial Interlinking. In *WWW '21: The Web Conference 2021*. 833–844.
- [26] Dimitris Papadias, Nikos Mamoulis, and Dimitris Meretakakis. 1998. Image Similarity Retrieval by Spatial Constraints. In *Proceedings of the 1998 ACM CIKM International Conference on Information and Knowledge Management, Bethesda, Maryland, USA, November 3-7, 1998*. ACM, 289–296.
- [27] Jignesh M. Patel and David J. DeWitt. 1996. Partition Based Spatial-Merge Join. In *Proceedings of the 1996 ACM SIGMOD International Conference on Management of Data, Montreal, Quebec, Canada, June 4-6, 1996*. ACM Press, 259–270.
- [28] PostGIS Development Team. 2023. *PostGIS*. PostGIS Development Group. <https://postgis.net/>
- [29] Franco P. Preparata and Michael Ian Shamos. 1985. *Computational Geometry - An Introduction*. Springer.
- [30] Ziyu Shang, Peng Wang, Yuzhang Liu, Jiajun Liu, and Wenjun Ke. 2023. ASKRL: An Aligned-Spatial Knowledge Representation Learning Framework for Open-World Knowledge Graph. In *The Semantic Web - ISWC 2023 - 22nd International Semantic Web Conference, Athens, Greece, November 6-10, 2023, Proceedings, Part I (Lecture Notes in Computer Science, Vol. 14265)*. Springer, 101–120.
- [31] Mohamed Ahmed Sherif, Kevin Dreßler, Panayiotis Smeros, and Axel-Cyrille Ngonga Ngomo. 2017. Radon - Rapid Discovery of Topological Relations. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence, February 4-9, 2017, San Francisco, California, USA*. AAAI Press, 175–181.
- [32] Yashbir Singh, Colleen M. Farrelly, Quincy A. Hathaway, Tim Leiner, Jaidip Jagtap, Gunnar E. Carlsson, and Bradley J. Erickson. 2023. Topological data analysis in medical imaging: current state of the art. *Insights into Imaging* 14, 1 (2023), 58. doi:10.1186/s13244-023-01413-w
- [33] Samridhhi Singla and Ahmed Eldawy. 2020. Raptor Zonal Statistics: Fully Distributed Zonal Statistics of Big Raster + Vector Data. In *2020 IEEE International Conference on Big Data (IEEE BigData 2020), Atlanta, GA, USA, December 10-13, 2020*. IEEE, 571–580.
- [34] Samridhhi Singla, Ahmed Eldawy, Tina Diao, Ayan Mukhopadhyay, and Elia Scudiero. 2021. Experimental Study of Big Raster and Vector Database Systems. In *37th IEEE International Conference on Data Engineering, ICDE 2021, Chanja, Greece, April 19-22, 2021*. IEEE, 2243–2248.
- [35] Samridhhi Singla, Ahmed Eldawy, Tina Diao, Ayan Mukhopadhyay, and Elia Scudiero. 2021. The Raptor Join Operator for Processing Big Raster + Vector Data. In *SIGSPATIAL '21: 29th International Conference on Advances in Geographic Information Systems, Virtual Event / Beijing, China, November 2-5, 2021*. ACM, 324–335.
- [36] A. Prasad Sistla, Clement T. Yu, and R. Haddad. 1994. Reasoning About Spatial Relationships in Picture Retrieval Systems. In *VLDB'94, Proceedings of 20th International Conference on Very Large Data Bases, September 12-15, 1994, Santiago de Chile, Chile*. Morgan Kaufmann, 570–581.
- [37] Christian Strobl. 2017. Dimensionally Extended Nine-Intersection Model (DE-9IM). In *Encyclopedia of GIS*. Springer, 470–476.
- [38] Dimitrios Tsitsigkos, Panagiotis Bouras, Konstantinos Lampropoulos, Nikos Mamoulis, and Manolis Terrovitis. 2024. Two-Layer Space-Oriented Partitioning for Non-Point Data. *IEEE Trans. Knowl. Data Eng.* 36, 3 (2024), 1341–1355.
- [39] Dimitrios Tsitsigkos, Panagiotis Bouras, Nikos Mamoulis, and Manolis Terrovitis. 2019. Parallel In-Memory Evaluation of Spatial Joins. In *Proceedings of the 27th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems, SIGSPATIAL 2019, Chicago, IL, USA, November 5-8, 2019*. ACM, 516–519.
- [40] Jia Yu, Zongsi Zhang, and Mohamed Sarwat. 2019. Spatial data management in apache spark: the GeoSpark perspective and beyond. *Geoinformatica* 23, 1 (2019), 37–78.
- [41] Rui Zhang, Hao Li, and Peng Yue. 2023. A Spatial-Aware Representation Learning Model for Link Completion in GeoKG: A Case Study on OpenStreetMap. In *11th International Conference on Agro-Geoinformatics, Agro-Geoinformatics 2023, Wuhan, China, July 25-28, 2023*. IEEE, 1–6.
- [42] Geraldo Zimbrão and Jano Moreira de Souza. 1998. A Raster Approximation For Processing of Spatial Joins. In *VLDB'98, Proceedings of 24th International Conference on Very Large Data Bases, August 24-27, 1998, New York City, New York, USA*. 558–569.