Indexing and Querying Vague Spatial Data Warehouses

Thiago Luís Lopes Siqueira[1,2,3], João Celso Santos de Oliveira[1], Valéria Cesário Times[4], Cristina Dutra de Aguiar Ciferri[5], Ricardo Rodrigues Ciferri[1]

[1] Federal University of São Carlos, Brazil
[2] Université Libre de Bruxelles, Belgium
[3] São Paulo Federal Institute of Education, Science and Technology (IFSP), Brazil
[4] Federal University of Pernambuco, Brazil
[5] University of São Paulo, Brazil
prof.thiago@ifsp.edu.br, joaocelso@comp.ufscar.br, vct@cin.ufpe.br, cdac@icmc.usp.br, ricardo@dc.ufscar.br

Abstract. A vague spatial data warehouse allows multidimensional queries with spatial predicates to support the analysis of business scores related to vague spatial data, crisp spatial data and conventional data. However, vague spatial data are often represented and stored as multiple geometries and impair the query processing performance. In this paper, we describe an index called VSB-index to improve the query processing performance in vague spatial data warehouses, focusing on range queries and vague regions. We also conduct an experimental evaluation using a real dataset, demonstrating that our VSB-index provided remarkable performance gains up to 94% over existing solutions.

Categories and Subject Descriptors: H.2 [**Database Management**]: Database Application—*Spatial databases and GIS*; H.2 [**Database Management**]: Physical design—*Access methods*

Keywords: Spatial Data Warehouses, Spatial Vagueness, Vague Regions, VSB-index

1. INTRODUCTION

Spatial vagueness is one kind of spatial data imperfection concerning the difficulty of distinguishing an object's shape from its neighborhood. Exact models represent vague spatial objects by reusing well-known crisp spatial data models, extending the theory of spatial data types and spatial predicates and implementing them as artifacts of vector-based geometries [Clementini and Di Felice 1996; Pauly and Schneider 2010]. For example, a vague region r might have a known extent and a broad boundary as a two-dimensional zone surrounding the known extent, instead of a one-dimensional line with minimal thickness [Clementini and Di Felice 1996]. Then, r consists of a pair of crisp regions: the *kernel* and the *conjecture*. The kernel is the known extent and the determinate part of r, while the conjecture is the broad boundary and vague part of r. The interior of the kernel is disjoint from the interior of the conjecture. Points encompassed by the kernel *certainly* belong to r, points encompassed by the conjecture *possibly* belong to r, and points not encompassed by the kernel and neither by the conjecture *do not* belong to r [Pauly and Schneider 2010].

Spatial vagueness affects several real world phenomena. Therefore, the design of vague spatial data warehouses (vague SDW), to allow multidimensional and spatial analysis of business scores regarding vague spatial data, has recently gained the attention of researchers. Not only vague spatial data are addressed, but also crisp spatial data and conventional spatial data. The data cube metaphor has

Authors thank CAPES, CNPq, FAPESP and FINEP. 1^{st}, 2^{nd}, and 3^{rd} authors are funded by CNPq, grants 229675/2013-1, 225742/2013-6 and 246263/2012-1, respectively. The 4^{th} author is funded by FAPESP, grant 2011/23904-7. Authors also thank Embrapa for providing the real dataset used in the experiments

been used in the conceptual design of vague SDW to enable flexible and multidimensional ways for representing, querying and aggregating vague spatial data [Siqueira et al. 2014]. The relational model has been reused to design logical schemas of vague SDW and store a subject-oriented, integrated, time-variant, voluminous, non-volatile and multidimensional database [Siqueira et al. 2012b]. In order to avoid an inappropriate use of the vague SDW in a decisional process, the risk associated to the inclusion of vague spatial data can be identified in advance and organized in tolerance levels before applying control strategies to obtain the definitive schema [Edoh-Alove et al. 2013].

In a vague SDW, a fact denotes the scores of business activities through numeric measures or spatial measures, while dimensions hold conventional attributes and crisp or vague spatial attributes that contextualize values of measures. Spatial range queries select specific parts of the vague spatial objects stored in the vague SDW, e.g. intersection range query (IRQ) [Siqueira et al. 2014]. In a previous work [Siqueira et al. 2013], we identified the lack of an index to process range queries against vague regions in vague SDWs. We also introduced and assessed the Vague Spatial Bitmap Index (VSB-index) that outperformed indices implemented by DBMSs and indices for crisp SDWs.

Still motivated by the challenge of improving the performance to process range queries against vague regions in vague SDWs, in this paper, we have kept the same direction by describing the VSB-index and assessing it through an experimental evaluation. Differently from our previous work [Siqueira et al. 2013], we have designed and queried a real dataset to demonstrate the high performance of the VSB-index and its applicability in real problem. In this sense, the novel experimental evaluation includes a set of representative tasks of typical workloads in the application domain, even though it has not used an existing application benchmark [Barbosa et al. 2009]. We have also reported novel results and discussed new insights on how designers can improve the performance of their systems with the VSB-index. This paper is organized as follows. Section 2 surveys related work, Section 3 describes the VSB-index, Section 4 tackles the design of a vague SDW with real agricultural data, Section 5 reports the experimental evaluation of the VSB-index and Section 6 concludes the paper.

2. RELATED WORK

In DWs, a bitmap join index created on the column C of a dimension table indicates the set of rows in the fact table to be joined with a certain value of C [O'Neil and Graefe 1995]. Although the bitmap join index avoids joining huge tables in DWs, it cannot solve spatial predicates.

The multistep spatial predicate resolution determines that progressive approximations and conservative approximations are used to identify answers of the spatial predicate in the filter step, to reduce the cost of the refinement step. A conservative approximation is a superset of the extent of the spatial object, while a progressive approximation is a subset of the extent of the spatial object [Brinkhoff et al. 1993]. Instead of reusing existing progressive approximations, in this paper, we describe a novel progressive approximation called MIP.

In crisp SDWs, the aR-tree [Papadias et al. 2001], the SB-index and the HSB-index [Siqueira et al. 2012a] are capable to process spatial predicates, conventional predicates, aggregation and sorting. They use a single conservative approximation for crisp spatial data: the MBR (minimum bounding rectangle). Conversely, our VSB-index uses both conservative and progressive approximations. The aR-tree and the HSB-index have hierarchical data structures and a tree-based search in the filter step (similar to the R-tree's [Guttman 1984]), while the SB-index has a sequential data structure and a sequential search in the filter step (similar to our VSB-index'). Conventional predicates, aggregation and sorting are processed by the aR-tree manipulating multidimensional arrays, while the SB-index, the HSB-index and our VSB-index reuse bitmap join indices to process them.

The vague R-tree [Petry et al. 2007] is an index for vague regions based on the R-tree, whose intermediate nodes maintain a pair of entries per cluster of vague regions. Entry O holds a MBR circumscribing the MBRs of the clustered vague regions, while entry I holds a MBR circumscribing

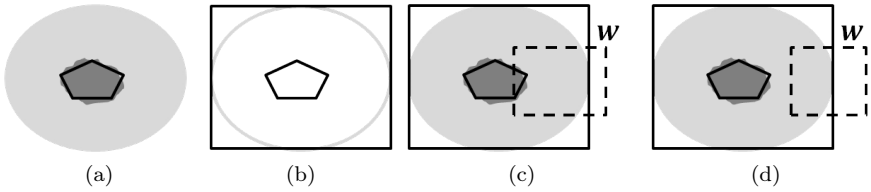

Fig. 1: Vague region, approximations and query: (a) The MIP5 on the kernel. (b) The MBR on outer boundary and the MIP5 on kernel. (c) A range query on $O_{\supseteq}K_{\subseteq}$. (d) Another range query on $O_{\supseteq}K_{\subseteq}$.

the MBRs of kernels of the clustered vague regions. Progressive approximations were not addressed and only algorithms for point queries were designed, differently from our VSB-index that uses progressive approximations and tackles range queries. Besides, the vague R-tree was not assessed through an experimental evaluation, differently from our VSB-index. Other existing indices that do not implement vague spatial objects as artifacts of vector-based geometries are beyond the scope of this paper.

3. THE VAGUE SPATIAL BITMAP INDEX

This section details the VSB-index. We prioritized a multistep resolution of the spatial predicate and created a specific progressive approximation to be used in the filter step to reduce the cost of the refinement step. We have chosen range queries as the spatial predicates to be supported by the VSB-index. Since queries issued on a vague SDW have not only spatial predicates, but also conventional predicates, aggregation and sorting, the latter three are processed by bitmap join indices that are often used in DWs. Section 3.1 introduces the progressive approximation MIP. Section 3.2 defines the data structure of the VSB-index. Section 3.3 focuses on the VSB-index query processing.

3.1 Maximum Area Inscribed Polygon

The Maximum Area Inscribed Polygon (MIP) is a progressive approximation consisting of a polygon with x vertices. The number of vertices is the suffix, e.g. MIP5 for $x = 5$. We define MIP to be applied specially on vague regions, to improve the resolution of spatial predicates in query processing. Figure 1a shows a vague region, its conjecture (light grey), its kernel (dark grey) and a MIP5 on its kernel (black contour). The outer boundary of the vague region encompasses the vague region and therefore a MIP5 on the kernel is also a subset of the outer boundary of the vague region, as shown in Figure 1b.

3.2 Data Structure

The VSB-index for vague regions has an array whose entries have the type *vrbitvector* (vague region bit-vector), comprising: (i) one key value pk; (ii) one mandatory conservative approximation $O_{\supseteq}$ on the outer boundary of the vague region; (iii) one optional progressive approximation $O_{\subseteq}$ on the outer boundary of the vague region; (iv) one optional conservative approximation $K_{\supseteq}$ on the kernel; and (v) one optional progressive approximation $K_{\subseteq}$ on the kernel. The notation considers the conservative approximation $\supseteq$ as a superset and the progressive approximation $\subseteq$ as a subset. All feasible configurations for the VSB-index are listed in Table I. The conservative approximation $O_{\supseteq}$ is mandatory to enable processing queries since the outer boundary of the vague region encompasses the vague region. Except $O_{\supseteq}$, the other approximations are optional and allow flexible data structures and query processing algorithms (Section 3.3). The VSB-index also has a bitmap join index created on the key values. Therefore, each entry of the array has a corresponding bitvector in the bitmap join index. The bitvector indicates the rows in the fact table that reference the entry's key value.

The size in bytes of a VSB-index entry is $s = sizeof(int)+sizeof(O_{\supseteq})+sizeof(O_{\subseteq})+sizeof(K_{\supseteq})+$

Table I: Entry size in bytes (s), number of entries per disk page (L) and number of disk pages to store the index file (A)

	$O_{\supseteq}O_{\subseteq}K_{\supseteq}K_{\subseteq}$	$O_{\supseteq}O_{\subseteq}K_{\supseteq}$	$O_{\supseteq}O_{\subseteq}K_{\subseteq}$	$O_{\supseteq}O_{\subseteq}$	$O_{\supseteq}K_{\supseteq}K_{\subseteq}$	$O_{\supseteq}K_{\supseteq}$	$O_{\supseteq}K_{\subseteq}$	$O_{\supseteq}$
s	228	148	196	116	148	68	116	36
L	35	55	41	70	55	120	70	227
A	5	4	5	3	4	3	3	2

$sizeof(K_{\subseteq})$. Each disk page with l bytes maintains $L = l$ DIV s index entries. Some unused bytes $U = c$ MOD L, where c is the cardinality of the indexed vague spatial attribute, are left between different disk pages to avoid fragmented entries and prevent two disk accesses to obtain a single entry. A header disk page stores metadata. Then, $A = 1 + c$ DIV $L + y$ disk pages are required to store the VSB-index. If c MOD $L = 0$ then $y = 0$, otherwise $y = 1$. Besides, A disk accesses are required to build the index file. Table I exemplifies values of s, L and A for MIP5, $l = 8192$ bytes and $c = 129$.

3.3 Query Processing

Range queries are supported by the VSB-index as follows. Let w be an iso-oriented rectangle called ad hoc spatial query window and S be a set of vague regions. An $IRQ_{poss}(w, S)$ concerns an intersection that is *possibly* true and retrieves vague regions in S whose outer boundary intersects w. Conversely, an $IRQ_{cert}(w, S)$ concerns an intersection that is *certainly* true and retrieves vague regions in S whose kernel intersects w. CRQ_{poss} and CRQ_{cert} are defined analogously for the relationship of containment. ERQ_{poss} and ERQ_{cert} are defined analogously for enclosure ("inside of").

The VSB-index query processing firstly performs a filter step as a sequential scan on the index file which requires A disk accesses, given by function $f1$ detailed in Figure 2a or function $f2$ detailed in Figure 2b. They produce candidates and answers of the spatial predicate and store them in their proper sets in the main memory, i.e. *setCandidates* and *setAnswers*, respectively.

Function $f1$ performs a sequential scan over the index file (lines 2-7), which retrieves each disk page (line 3) and temporarily stores it in the main memory (line 4). Function *get* obtains the conservative approximation of every entry transferred to main memory (line 6). Such conservative approximation is $O_{\supseteq}$ or $K_{\supseteq}$, depending on the parameter passed, and is tested against the ad hoc spatial query window (line 6). If the spatial relationship is satisfied, the entry's primary key value is appended to a set (line 7). Finally, the index file is closed (line 8). The aforementioned set might be the set of candidates or the set of answers, depending on the parameter passed.

To identify answers already in the filter step, function $f2$ performs a sequential scan that firstly tests the conservative approximation and secondly tests the progressive approximation. For each entry (lines 5-10), if the spatial relationship is satisfied for both the conservative and progressive approximations, the entry is considered as an answer and its primary key value is stored in the set of answers (lines 6-8). However, if only the conservative approximation satisfies the spatial relationship, the entry is considered as a candidate and its primary key value is stored in the set of candidates (line 10). According to the calls, $f2$ is particularly useful for IRQ_{poss} when $O_{\supseteq}$ and $O_{\subseteq}$ are available, and for IRQ_{cert} when $K_{\supseteq}$ and $K_{\subseteq}$ are available. Also, $K_{\subseteq}$ can be used to fetch results when querying IRQ_{poss}, as well as $O_{\supseteq}$ can be used to indicate candidates when querying IRQ_{cert}.

The filter step is a call to $f1$ or $f2$ based on a decision regarding the spatial predicate to evaluate and which approximations are available among $O_{\supseteq}$, $O_{\subseteq}$, $K_{\supseteq}$ and $K_{\subseteq}$. In this paper, we focus on IRQ_{cert} and IRQ_{poss} which are solved by $f1$ or $f2$. The 16 calls to process IRQ_{cert} and IRQ_{poss} are listed in Table III. For instance, configuration $O_{\supseteq}K_{\subseteq}$ calls $f2$ and adds the vague region shown in Figure 1c to the set of answers already in the filter step, for both IRQ_{cert} and IRQ_{poss}.

After the filter step, the refinement step is performed using the DBMS and its results are recorded in *setAnswers*. A key-matching in *setAnswers* produces a string with a conventional predicate based

(a)

```
Algorithm 1: f1(R, w, conservative, set, idx, L)
Input: parameters described in Table II
Declarations: page, array
Output: a set of candidates or a set of answers
of the spatial predicate
1  open (idx)
2  while not (eof(idx)) do
3     read (idx, page)
4     copy (page, array)
5     for i ← 0 to L do
6        if R(w, get(array[i],conservative))
7           append(set, array[i].pk)
8  close(idx)
```

(b)

```
Algorithm 2: f2(R, w, conservative, progressive,
setCandidates, setAnswers, idx, L)
Input: parameters described in Table II
Declarations: page, array
Output: the set of candidates and the set of
answers of the spatial predicate
1  open (idx)
2     while not (eof(idx)) do
3       read (idx, page)
4       copy (page, array)
5       for i ← 0 to L do
6         if R(w, get(array[i],conservative))
7            if R(w, get(array[i],progressive))
8             append(setAnswers, array[i].pk)
9            else
10            append(setCandidates, array[i].pk)
11 close(idx)
```

Fig. 2: Algorithms: (a) Function $f1$. (b) Function $f2$.

Table II: Parameters of Algorithms 1 and 2

Parameter	Description
conservative	Indicator for $O_{\supseteq}$ or $K_{\supseteq}$
conservativeK	Indicator for $K_{\supseteq}$
conservativeO	Indicator for $O_{\supseteq}$
idx	The VSB-index file
L	The maximum number of index entries that a disk page can hold
progressive	Indicator for $O_{\subseteq}$ or $K_{\subseteq}$
progressiveK	Indicator for $K_{\subseteq}$
progressiveO	Indicator for $O_{\subseteq}$
pk	The primary key attribute of *table*
R	The spatial relationship (intersection, containment or enclosure)
setAnswers	The set of answers of the spatial predicate
setCandidates	The set of candidates (possible answers) of the spatial predicate
table	The vague spatial dimension table queried
vsa	The vague spatial attribute of *table*
w	The ad hoc spatial query window

on primary key values of the vague regions that satisfy the spatial predicate. Such string replaces the spatial predicate of the query submitted to the vague SDW. Finally, the rewritten query is solved by efficient bitmap join indices that avoid joining huge SDW tables and provide the query answer.

4. A VAGUE SPATIAL DATA WAREHOUSE ON CITRUS GREENING INFECTION

Greening [1] is a serious disease that infects citrus and impairs the industry. It is caused by a bacterium transmitted by an insect. As there is not a cure so far, its control is done by visual inspection and immediate eradication of the infected plant by the roots. Temporal and spatial patterns of distribution of greening in the field at different scales, as plots and cities, are crucial to reduce the rate of failures in visual inspections. The infection is examined at various spatial levels, e.g. immediately adjacent trees within and across row, and trees estimated to be 25 to 30 meters far from an infected tree.

A real dataset regarding citrus greening infection was provided by the Brazilian Agricultural Research Corporation (Embrapa). It comprises data collected in the field in 13 different months regarding a citrus plot with approximately 9,000 trees. Every area infected by greening was modeled as a vague

[1] http://www.plantmanagementnetwork.org/pub/php/review/2007/huanglongbing/

Table III: Calls made to functions $f1$ and $f2$ by configuration of the VSB-index

Configuration	Function calls of IRQ_{cert} and IRQ_{poss} (R=intersection)
$O_{\supseteq}O_{\subseteq}K_{\supseteq}K_{\subseteq}$	IRQ_{cert}: f2 (R, w, conservativeK, progressiveK, setCandidates, setAnswers, idx, L) IRQ_{poss}: f2 (R, w, conservativeO, progressiveO, setCandidates, setAnswers, idx, L)
$O_{\supseteq}O_{\subseteq}K_{\supseteq}$	IRQ_{cert}: f1(R, w, conservativeK, setCandidates, idx, L) IRQ_{poss}: f2 (R, w, conservativeO, progressiveO, setCandidates, setAnswers, idx, L)
$O_{\supseteq}O_{\subseteq}K_{\subseteq}$	IRQ_{cert}: f2 (R, w, conservativeO, progressiveK, setCandidates, setAnswers, idx, L) IRQ_{poss}: f2 (R, w, conservativeO, progressiveO, setCandidates, setAnswers, idx, L)
$O_{\supseteq}O_{\subseteq}$	IRQ_{cert}: f1(R, w, conservativeO, setCandidates, idx, L) IRQ_{poss}: f2 (R, w, conservativeO, progressiveO, setCandidates, setAnswers, idx, L)
$O_{\supseteq}K_{\supseteq}K_{\subseteq}$	IRQ_{cert}: f2 (R, w, conservativeK, progressiveK, setCandidates, setAnswers, idx, L) IRQ_{poss}: f2 (R, w, conservativeO, progressiveK, setCandidates, setAnswers, idx, L)
$O_{\supseteq}K_{\supseteq}$	IRQ_{cert}: f1(R, w, conservativeK, setCandidates, idx, L) IRQ_{poss}: f1(R, w, conservativeO, setCandidates, idx, L)
$O_{\supseteq}K_{\subseteq}$	IRQ_{cert}: f2 (R, w, conservativeO, progressiveK, setCandidates, setAnswers, idx, L) IRQ_{poss}: f2 (R, w, conservativeO, progressiveK, setCandidates, setAnswers, idx, L)
$O_{\supseteq}$	IRQ_{cert}: f1(R, w, conservativeO, setCandidates, idx, L) IRQ_{poss}: f1(R, w, conservativeO, setCandidates, idx, L)

region. Figure 3a shows 2 vague regions a_1 and a_2. The kernel was the extent where trees were infected and eradicated. Conversely, the conjecture was the broad boundary where the insect possibly transmitted the bacterium to trees that were not eradicated, but that became suspicious.

Each tree was a point with a status of infected or healthy. The kernel of an infected area was a buffer applied on the point of each infected tree (e.g. a_1 in Figure 3a). A buffer did not intersect any other tree (infected or healthy). Since infection might happen in neighbor trees, if two or more buffers intersected and did not cover any healthy tree, then such buffers were merged and considered as a single kernel, where two or more trees were eradicated (e.g. a_2 in Figure 3a). Furthermore, the conjecture of each infected area was outlined by firstly fetching all trees within 25 meters from each infected tree. Secondly, a convex hull containing each infected tree and its neighbors was built (e.g. a_1 in Figure 3a). The convex hulls built for two or more different infected trees were also merged whether their corresponding kernels had already been merged previously (e.g. a_2 in Figure 3a).

The vague SDW depicted in Figure 3b was built according to existing guidelines [Siqueira et al. 2012b]. *GreeningInfection* is a fact table referencing dimension tables and holding the numeric measure of quantity of eradicated plants. The cited vague regions were stored in *InfectedArea* that is a vague spatial measure pushed in a vague spatial dimension table with the vague spatial attribute *infectedarea_vgeo* of type multipolygon. *Plot* is a crisp spatial dimension table with the crisp spatial attribute *plot_geo* of type polygon. *Inspector* and *Date* are conventional dimension tables. Instead of comparing a vague region to another, which has been widely tackled in the literature [Clementini and Di Felice 1996; Pauly and Schneider 2010], the user needs to retrieve areas certainly infected (IRQ_{cert}) and areas possibly infected (IRQ_{poss}), according to the intersection against an ad hoc rectangular query window w such as that shown in Figure 3a. A tipical decisional query is shown in Figure 4, where IRQ_{cert} or IRQ_{poss} is chosen as spatial predicate.

5. EXPERIMENTAL EVALUATION OF THE VSB-INDEX

This section details the performance evaluation of the VSB-index. Instead of reusing an existing application benchmark, we have designed a set of representative tasks of typical workloads in the domain of greening infection, based on Section 4. We have tackled the spatial predicate resolution since it was identified as a bottleneck to process queries involving vague regions in vague SDWs [Siqueira et al. 2013]. Section 5.1 addresses the experimental setup. Results for IRQ_{poss} and IRQ_{cert} are reported in Sections 5.2 and 5.3, respectively. Section 5.4 has final remarks.

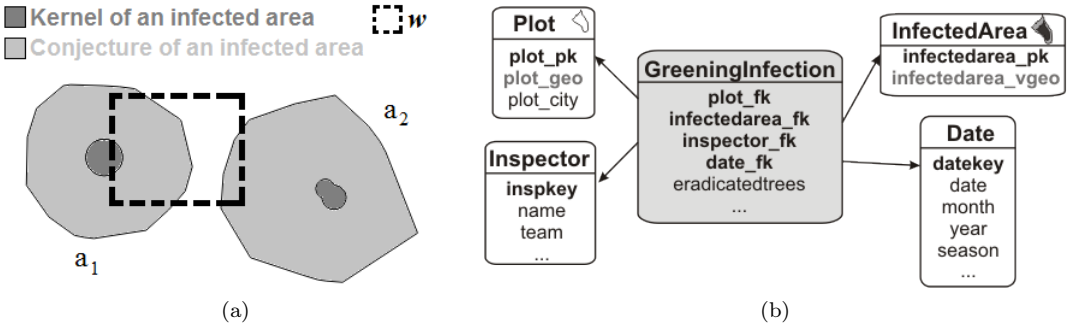

Fig. 3: A vague SDW on greening infection: (a) vague spatial data distribution (b) vague SDW schema

```
SELECT team, year, SUM(eradicatedtrees) FROM Inspector, Date, InfectedArea, GreeningInfection
WHERE inspkey=inspector_fk AND datekey=date_fk AND infectedarea_fk=infectedarea_pk AND
team='XY' AND [IRQ_cert|IRQ_poss](appliedarea_vgeo,w) GROUP BY team, year ORDER BY team, year
```

Fig. 4: Querying the vague SDW shown in Figure 3b

5.1 Experimental Setup

The workbench comprised the vague SDW shown in Figure 3b. The attribute *infectedarea_geo* had 129 distinct infected areas. Both *InfectedArea* and *GreeningInfection* stored 129 rows. The workload was based on the spatial predicate of the query shown in Figure 4, as follows. Given a dataset with cardinality c, and a spatial query window w that retrieves k objects from the dataset, the selectivity was $k \div c$. We used the following selectivities for IRQ$_{poss}$: 0, 0.01, 0.02, 0.03, 0.04, 0.05, 0.10, 0.50 and 1.00. The null selectivity represents the selection of zero elements of the dataset. Low selectivities under 0.10 denote the user selecting a constrained extent of interest. Moderate selectivities of 0.10 and 0.50 concern the user selecting a wider extent of interest. The high selectivity of 1.0 retrieves the whole dataset, e.g. when it must be displayed. Moderate and high selecitivies were acceptable since the data volume of vague regions was low. For each selectivity, distinct spatial query windows were used in each of the 10 consecutive IRQ$_{poss}$ that were issued. System cache was flushed between executions. We gathered the average elapsed time and the average number of candidates for each configuration of the VSB-index. The time reduction measured how much a configuration was more efficient than another. The same procedure was followed to evaluate IRQ$_{cert}$.

The platform was a computer with a 3.2 GHz Pentium D processor, 8 GB of main memory, a 7200 RPM SATA 320 GB hard disk with 8 MB of cache, Linux CentOS 6.4, PostgreSQL 9.2.2 and PostGIS 2.0.1. All feasible configurations of the VSB-index were implemented using MBR as conservative approximation and MIP5 as progressive approximation - 5 vertices by analogy with 5C [Brinkhoff et al. 1993]. They were implemented in C/C++ and the disk page size was set to 8 KB. MIP5 was built using the CGAL, Computational Geometry Algorithms Library (http://www.cgal.org) version 4.0.2 and the method CGAL::maximum_area_inscribed_k_gon_2. The method uses monotone matrix search [Aggarwal et al. 1987] and has a worst case running time of $O(x \times n + n \times logn)$, where n is the number of vertices provided as input and x is the number of vertices of the MIP.

5.2 IRQposs

Figure 5a reports the elapsed time to process IRQ_{poss}, while Figure 5b details the average number of candidates processed in the refinement step by each configuration. High average number of candidates as 65 and 129 were omitted (configuration $O_{\supseteq}$ and selectivities 0.50 and 1.00, respectively). Note that:

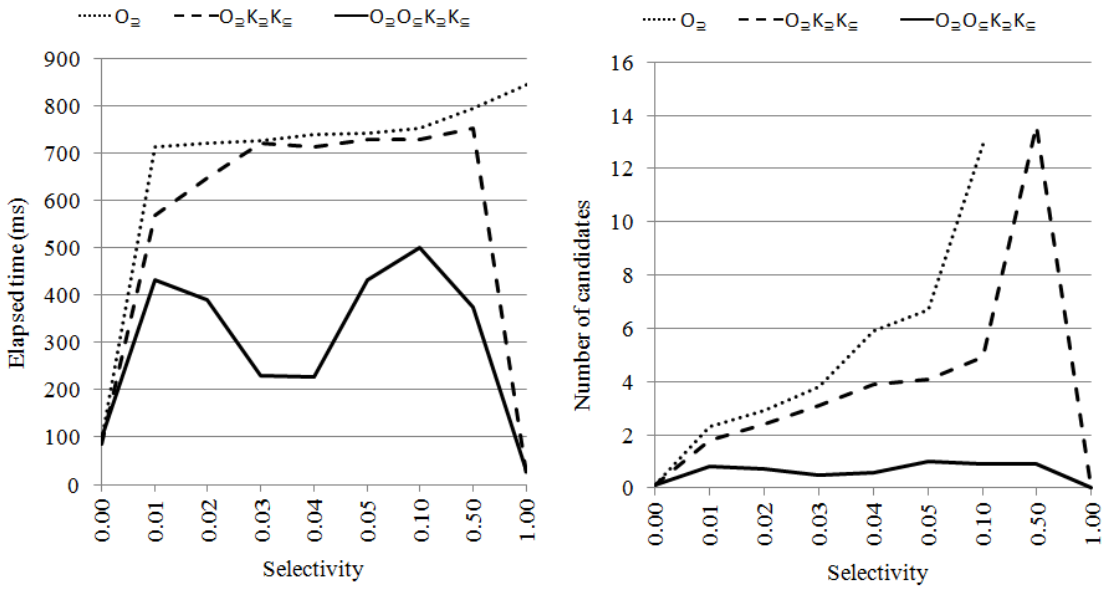

Fig. 5: Results for IRQposs: (a) Average elapsed time. (b) Average number of candidates.

—Configurations $O_{\supseteq}$ and $O_{\supseteq}O_{\subseteq}$, and the SB-index had similar performances and were reported together.

—Configurations $O_{\supseteq}K_{\supseteq}K_{\subseteq}$ and $O_{\supseteq}K_{\subseteq}$ had similar performances and were reported together.

—Configurations $O_{\supseteq}O_{\subseteq}K_{\supseteq}K_{\subseteq}$, $O_{\supseteq}K_{\supseteq}$, $O_{\supseteq}O_{\subseteq}K_{\supseteq}$ and $O_{\supseteq}O_{\subseteq}K_{\subseteq}$ had similar performances and were reported together.

Every configuration took time to process queries with selectivity 0, executing the filter step and the refinement step whether there were candidates. Regarding selectivities between 0.01 and 0.50, the average number of candidates processed by each configuration determined its performance, as follows.

Configurations that did not hold MIP5, as $O_{\supseteq}$, required longer elapsed times independently of the selectivity. Their filter steps were not able to identify answers of IRQ_{poss} due to the absence of a progressive approximation. Then, more candidates were processed by the refinement step, increasing its cost and causing low performance.

Configurations that held a single MIP5 on the kernel, as $O_{\supseteq}K_{\supseteq}K_{\subseteq}$, were able to identify answers of the IRQ_{poss} by calling the function $f2$ and accessing the MIP5 on the kernel. However, not many answers were identified since the MIP5 on the kernel might encompass only a small portion of the vague region, as shown in Figure 1d. As a result, their average numbers of candidates were not drastically reduced, the refinement step was costly and their elapsed times were not decreased.

Shortert elapsed times were spent by configurations with MIP5 on the outer boundary of the vague region, as $O_{\supseteq}O_{\subseteq}K_{\supseteq}K_{\subseteq}$. They had low average numbers of candidates, even for greater selectivities. This trend, however, did not occur in the other configurations. Due to processing fewer candidates in the refinement step, configuration $O_{\supseteq}O_{\subseteq}K_{\supseteq}K_{\subseteq}$ imposed a time reduction of at least 33.36% and at most 69.19% over SB-index (configuration $O_{\supseteq}$) for selectivities 0.10 and 0.04, respectively.

Finally, regarding the selectivity 1.00, all configurations holding at least one MIP5 had an empty set of candidates to process in the refinement step, since all answers were identified already in the filter step. As a result, they required shorter elapsed times.

5.3 IRQcert

Figure 6a reports the elapsed time to process IRQ_{cert}, while Figure 6b details the average number of candidates processed in the refinement step by each configuration. High average number of candidates as 65 and 129 were omitted (configuration $O_{\supseteq}$ and selectivities 0.50 and 1.00, respectively). Note that:

—Configurations $O_{\supseteq}$, $O_{\supseteq}O_{\subseteq}$, $O_{\supseteq}K_{\supseteq}$, $O_{\supseteq}O_{\subseteq}K_{\supseteq}$, and the SB-index had similar performances and were reported together.
—Configurations $O_{\supseteq}K_{\subseteq}$ and $O_{\supseteq}O_{\subseteq}K_{\subseteq}$ had similar performances and were reported together.
—Configurations $O_{\supseteq}O_{\subseteq}K_{\supseteq}K_{\subseteq}$ and $O_{\supseteq}K_{\supseteq}K_{\subseteq}$ had similar performances and were reported together.

Every configuration took time to process queries with selectivity 0, executing the filter step and the refinement step whether there were candidates. Regarding selectivities between 0.01 and 0.50, the average number of candidates processed by each configuration determined its performance, as follows.

Configurations that did not hold MIP5 on the kernel, as $O_{\supseteq}$, spent longer elapsed times independently of selectivity. Their filter steps were not able to identify answers of IRQ_{cert} due to the absence of a progressive approximation on the kernel. Then, more candidates were processed by the refinement step, increasing its cost and causing low performance.

Configurations that held MIP5 on the kernel but did not hold MBR on the kernel, as $O_{\supseteq}K_{\subseteq}$, accessed the MBR on the outer boundary of the vague region in their filter steps. Since the MBR on the outer boundary might cover a wider extent than the extent covered by the MIP5 on the kernel (e.g. Figure 1d), these configurations were not able to diminish the set of candidates. Therefore, the costly refinement step determined a low performance.

Configurations that held both MBR and MIP5 on the kernel, as $O_{\supseteq}O_{\subseteq}K_{\supseteq}K_{\subseteq}$, required shorter elapsed time to process IRQ_{cert} than the other configurations. Their average number of candidates was low due to the use of MIP5 on the kernel. Therefore, their refinement steps were less costly and the performance was benefited. The configuration $O_{\supseteq}O_{\subseteq}K_{\supseteq}K_{\subseteq}$ provided a time reduction of at least 84.22% and at most 94.77% over SB-index ($O_{\supseteq}$), for selectivities 0.05 and 0.02, respectively.

Configurations with a MIP5 on the kernel had an empty set of candidates to process in the refinement step for selectivity 1.00. As all answers were identified in the filter step, the elapsed times were shorter.

5.4 Final Remarks

In the previous sections, we identified that the VSB-index outperformed the SB-index. Besides, the VSB-index configuration $O_{\supseteq}O_{\subseteq}K_{\supseteq}K_{\subseteq}$ achieved the best performance for both IRQ_{poss} and IRQ_{cert}. Results also revealed that, to achieve a high performance to resolve IRQ_{poss} and IRQ_{cert} against vague regions, it is essential to reduce the number of candidates to be processed in the refinement step. In this sense, the use of MIP5 as progressive approximation was beneficial.

6. CONCLUSIONS AND FUTURE WORK

In this paper, we detailed the VSB-index to process multidimensional queries extended with intersection range queries against vague regions in vague SDWs. We assessed the VSB-index through an experimental evaluation using a vague SDW designed on a real dataset, to demonstrate the applicability of the VSB-index in a real problem. Results corroborated the efficiency of the VSB-index that had remarkable performance gains up to 94% over existing solutions. Results also reinforced the importance of an index for vague SDW. As future work, we intend to extend the VSB-index to index other data types as vague points and vague lines [Pauly and Schneider 2010], and to enable vague SOLAP operations as roll-up and drill-down [Siqueira et al. 2014].

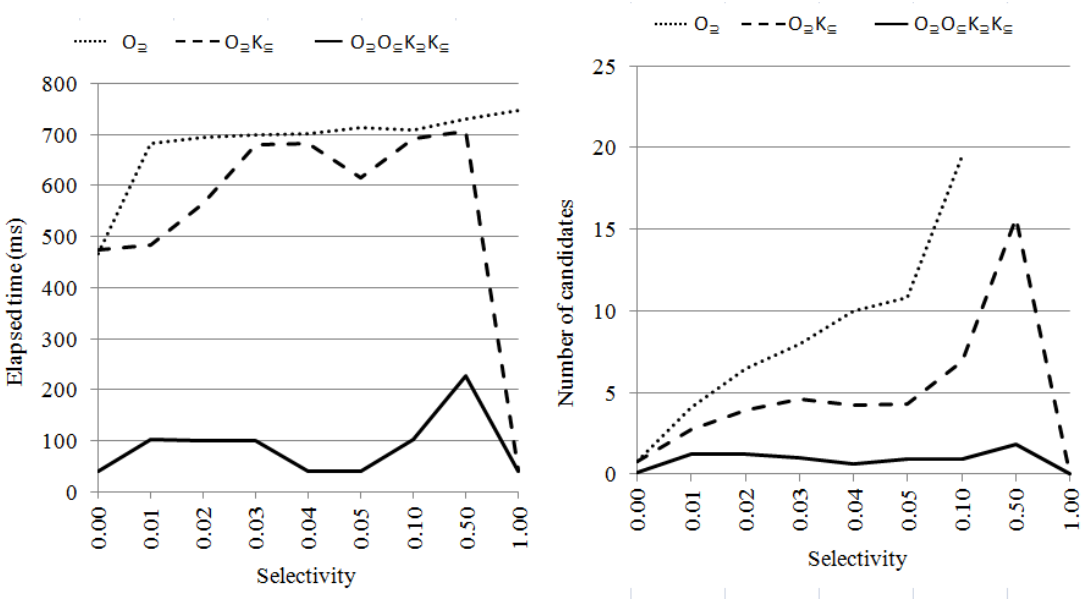

Fig. 6: Results for IRQcert: (a) Average elapsed time (ms). (b) Average number of candidates.

REFERENCES

Aggarwal, A., Klawe, M., Moran, S., Shor, P., and Wilber, R. Geometric Applications of a Matrix-Searching Algorithm. *Algorithmica* 2 (2): 195–208, 1987.

Barbosa, D., Manolescu, I., and Yu, J. X. Application Benchmark. In *Encyclopedia of Database Systems*, L. Liu and M. T. Özsu (Eds.). Springer US, pp. 99–100, 2009.

Brinkhoff, T., Kriegel, H.-P., and Schneider, R. Comparison of Approximations of Complex Objects used for Approximation-Based Query Processing in Spatial Database Systems. In *Proceedings of the IEEE International Conference on Data Engineering*. Vienna, Austria, pp. 40–49, 1993.

Clementini, E. and Di Felice, P. An Algebraic Model for Spatial Objects with Indeterminate Boundaries. In *Geographic Objects with Indeterminate Boundaries*. Vol. 2. Taylor & Francis London, pp. 155–169, 1996.

Edoh-Alove, E., Bimonte, S., Pinet, F., and Bédard, Y. Exploiting Spatial Vagueness in Spatial OLAP: Towards a new hybrid risk-aware design approach. In *Proceedings of the AGILE Conference*. Leuven, Belgium, pp. 1–4, 2013.

Guttman, A. R-trees: A dynamic index structure for spatial searching. *SIGMOD Record* 14 (2): 47–57, 1984.

O'Neil, P. and Graefe, G. Multi-table Joins Through Bitmapped Join Indices. *SIGMOD Record* 24 (3): 8–11, 1995.

Papadias, D., Kalnis, P., Zhang, J., and Tao, Y. Efficient OLAP Operations in Spatial Data Warehouses. In *Proceedings of the International Symposium on Advances in Spatial and Temporal Databases*. Redondo Beach, California, USA, pp. 443–459, 2001.

Pauly, A. and Schneider, M. VASA: An algebra for vague spatial data in databases. *Information Systems* 35 (1): 111–138, 2010.

Petry, F. E., Ladner, R., and Somodevilla, M. Indexing Implementation for Vague Spatial Regions with R-trees and Grid Files. In *Geographic Uncertainty in Environmental Security*, A. Morris and S. Kokhan (Eds.). NATO Science for Peace and Security Series C: Environmental Security. Springer Netherlands, pp. 187–199, 2007.

Siqueira, T. L., Ciferri, C. D. A., Times, V. C., and Ciferri, R. R. The SB-index and the HSB-index: Efficient Indices for Spatial Data Warehouses. *Geoinformatica* 16 (1): 165–205, 2012a.

Siqueira, T. L. L., Ciferri, C. D. A., Times, V. C., and Ciferri, R. R. Towards Vague Geographic Data Warehouses. In *Proceedings of the GIScience International Conference*. Columbus, Ohio, USA, pp. 173–186, 2012b.

Siqueira, T. L. L., Ciferri, C. D. A., Times, V. C., and Ciferri, R. R. Modeling Vague Spatial Data Warehouses using the VSCube Conceptual Model. *GeoInformatica* 18 (2): 313–356, 2014.

Siqueira, T. L. L., Oliveira, J. C. S., Times, V. C., Ciferri, C. D. A., and Ciferri, R. R. Indexing Vague Regions in Spatial Data Warehouses. In *Proceedings of the Brazilian Symposium on GeoInformatics*. Campos do Jordão, São Paulo, Brazil, pp. 158–169, 2013.