Approximations for a Multi-Step Processing of Spatial Joins

Thomas Brinkhoff and Hans-Peter Kriegel

Institute for Computer Science, University of Munich
Leopoldstr. 11 B, D-80802 München, Germany
e-mail: {brink, kriegel}@informatik.uni-muenchen.de

Abstract. The basic concept for processing spatial joins consists of two steps: First, the spatial join is performed on the minimum bounding rectangles of the objects by using a spatial access method. This step provides a set of candidates which consists of answers (hits) and non-answers (false hits). In the second step, the exact geometry of the candidates is transferred from secondary storage into main memory and is tested against the join predicate. This step is called refinement step. It causes the main cost for computing a spatial join. In this paper, we introduce an additional filter step in order to reduce the cost of the refinement step. In this filter step more sophisticated approximations are used to identify hits as well as to filter out false hits from the set of candidates. For this purpose, we investigate various types of conservative and progressive approximations. The performance of the approximation approach is evaluated with data sets from real cartographic applications. The results show that this approach considerably reduces the total execution time of the spatial join.

1 Introduction

Recently, several *spatial database systems (spatial DBSs)*, particularly designed for organizing spatial data of a geographic information system (GIS), have been developed for applications such as cartography, environmental science and geography. For these applications, the data volume is extremely high, the spatial objects show a very complex structure and the computation of spatial operators is time-intensive. Therefore, the requirements on a spatial DBS are particularly related to efficient query processing.

A *spatial object* consists of (at least) one spatial attribute that describes the geometry of the object. A typical spatial query is the *window query* which computes all objects of a given set of spatial objects *(map)* whose geometric component overlaps with a given rectilinear query rectangle. In contrast to a window query, the *spatial join* is defined on two maps. The spatial join computes a subset of the Cartesian product. It combines spatial objects according to their geometric attributes, i.e. these attributes have to fulfill a *spatial predicate*. One of the most frequently used spatial joins is the *intersection join* where pairs of objects are computed whose geometry intersects.

In order to improve the performance of the spatial join, we favor to process the join in three steps. Each step uses a different representation of the objects:

- First, instead of using an exact representation, the *minimum bounding rectilinear rectangle (MBR)* is used for computing the so-called *MBR-join*. This step returns a *candidate set* that contains answers *(hits)* and additionally elements of the Cartesian product which do not fulfill the join predicate *(false hits)*. This step should be supported by *spatial access methods (SAMs)*. Recently, several papers on MBR-joins using SAMs were published, e.g. [BHF 93], [Gün 93], [BKS 93a], and [LR 94].
- In the second step *(approximation step)*, more accurate approximations are exploited for filtering out false hits from the candidate set. Moreover, approximations can also be used to identify hits without accessing the exact representation of the

spatial objects. The more candidates are identified as hit or false hit, the less expensive becomes the last step.
- Finally, in the third step (*refinement step* [Ore 89]), all remaining members of the candidate set are examined for being answers to the spatial join. This step requires access to the exact representation of the spatial objects. In the context of spatial join processing, the refinement step is discussed e.g. in [BG 90] and [BKSS 94].

In this paper, the *approximation step* is investigated for intersection joins. The approximation of objects based on a raster representation was already examined in [OM 88] using sets of z-values and in [Sam 90] using PR-quadtrees. In contrast to these approaches, we assume that the geometry of the approximations as well as the objects themselves are represented by polygonal areas; i.e. the data model is the same for the approximation and for the exact geometry. Obviously, there exists a great variety of such approximations and the question arises what kind of approximations are most suitable to query processing and in particular to spatial join processing.

The paper is organized as follows. In section 2, several approximations are introduced for performing the approximation step. The impact of using additional approximations on the performance of the MBR-join is investigated in section 3. The total performance improvements are discussed in the fourth section. Section 5 concludes the paper and gives an outlook to future work.

2 Approximations

A spatial access method organizes the spatial objects according to a geometric key. The *minimum bounding rectangle (MBR)* is the most popular geometric key. Using the MBR, the complexity of an object is reduced to four parameters where the most important features of the object (position and extension) are maintained. A further important advantage of the MBR is the fast execution of spatial operations like the point-in-MBR test or the test for intersection. Consequently, the MBR is widely used.

Real cartography objects are only roughly approximated by MBRs. As a consequence, the exact object representation is often unnecessarily loaded into main memory and tested with costly computational geometry algorithms. In order to investigate the potential of using additional approximations, we performed empirical tests with real cartography data. As basic data we used two maps: *Europe* (counties in West Europe) and *BW* (municipalities in Baden-Württemberg). In order to obtain suitable test series, we pursue two strategies: Performing strategy *A*, the original map is joined with a second map which is generated by shifting the objects of the first map in x- and y-direction. The test series generated following strategy *A* are called *Europe A* and *BW A*. The other strategy *B* generates two maps per test series randomly shifting and rotating the objects of the original map. Additionally, the polygons are scaled in such a way that the sum of the object areas is equal to the area of the data space. The test series generated following strategy *B* are called *Europe B* and *BW B*. Table 1 depicts their characteristics.

test series	# objects per map	# points per polygon	# intersecting MBRs	# hits	# false hits
Europe A	810	84	1858	1273	585
Europe B	810	84	4816	3203	1613
BW A	374	527	2253	1504	749
BW B	374	527	2562	1684	878

Tab. 1. Test data

Table 1 demonstrates that about one third of the object pairs computed by the MBR-join are false hits, i.e. although the MBRs of the objects intersect, the objects themselves do not intersect. The reason for the high number of false hits is the rough approximation of the MBRs. Consequently, there is a great potential to improve the processing of spatial joins by using approximations which describe the spatial objects more exactly than the MBR. In the next subsections, we present a detailed investigation of such approximations for intersection joins.

2.1 Reducing the number of false hits

In order to identify more false hits, we examine several *conservative approximations*. An approximation is called conservative iff each point inside the contour of the original object is also in the conservative approximation. The basic idea is to perform the test for the join predicate on additional conservative approximations. This is expected to be much cheaper than reading the complete objects from secondary storage and performing the tests on their exact geometry.

We investigated five convex conservative approximations: the *rotated minimum bounding rectangle (RMBR)*, the *convex hull (CH)*, the *minimum bounding m-corner (m-C)*, the *minimum bounding circle (MBC)*, and the *minimum bounding ellipse (MBE).* Figure 1 visualizes the selected approximations using Great Britain as an example. These approximations differ especially in their accuracy and number of parameters which is given in brackets. The convex hull has on the average the highest storage requirement (and best accuracy) and the circle the lowest storage requirement [BKS 93b].

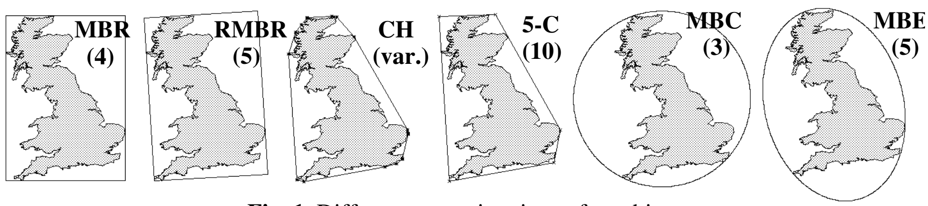


Fig. 1. Different approximations of an object

Quality of filtering using conservative approximations

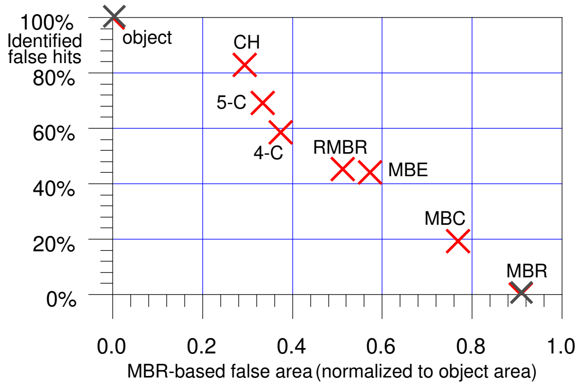
As mentioned before, if conservative approximations of two objects do not intersect, it follows that also the objects do not intersect. For intersection joins, we expect to identify considerably more false hits by using additional approximations. This expectation is confirmed by the test results depicted in table 2. It is assumed that the approximation is stored *in addition to* the MBR.

MBC	RMBR	MBE	4-C	5-C	CH
17.7	40.9	43.5	55.3	67.6	81.3

Tab. 2. Percentage of false hits identified by additional approximations (average on all test series)

The columns give the percentage of identified false hits. The results demonstrate that a high percentage of false hits can be identified by using these approximations. For example, the 5-C detects about 68% of the false hits. Thus, only 32% of the non-intersecting objects whose MBRs intersect, must be transferred into main memory and investigated in the exact geometry test.

In figure 2, we depict the quality of an approximation and the percentage of identified false hits for the *Europe B* test series. For defining a quality measure for approximations, we consider the *false area* which is the difference between the area of an object and of its approximation. Because the approximation is stored additionally to the MBR, we use the *MBR-based false area* as a measure for the *approximation quality*. That means, we first compute the intersection I of the approximation and the MBR and then the false area between the object and the intersection I. This is necessary because the approximation may cover parts of the data space which are not covered by the MBR. For comparability, the MBR-based false area is normalized to the object area.



MBR-based false area normalized to the object area (fa_n):

$$fa_n(obj) = \frac{\text{area}(I) - \text{area}(obj)}{\text{area}(obj)}$$

$$I = \text{appr}(obj) \cap \text{MBR}(obj)$$

obj:	object
appr(*obj*):	its approximation
MBR(*obj*)	its MBR

Fig. 2. Dependency between the approximation quality and the percentage of identified false hits

The results show that the more parameters are available for the representation of an additional approximation, the better is its approximation quality. The area of the 5-corner is nearly as accurate as the area of the convex hull. The MBR-based false areas of a RMBR and of an MBE are also considerably less than the one of the MBR. Considering in the MBR, the MBC, the RMBR, the 4-C, and the object itself, we recognize that the dependency is almost linear. However, the deviation of the 5-C, the MBE, and the convex hull demonstrates that not only the false area, but also the adaptability to the object is an important property of an approximation for identifying non-intersecting objects.

Because of its good approximation quality, the convex hull shows the best results. However, its number of vertices varies extremely and its average storage requirements are by factors higher than the requirements of the other approximations: the CH needs on the average 26 parameters for *Europe* and 46 for *BW* whereas the 5-C requires only 10 parameters. Therefore, it is problematic to store the CH in the data pages of a spatial access method which requires high numbers of entries within its data pages in order to achieve a good query performance. Overall, the 5-corner seems to be a good compromise: it needs much less storage, but it has a very good approximation quality and detects about two thirds of the non-intersecting object pairs delivered from the MBR-join.

2.2 Identifying hits in geometric filtering

In the previous subsection, we strived for reducing the number of false hits. As depicted in table 1, the number of hits clearly exceeds the number of false hits. After avoiding a large percentage of false hits, the ratio false hits to hits approaches 1 to 5. Therefore, it is not reasonable to invest more time in identifying further false hits. Instead, we examine techniques which identify intersecting objects *(hits)* without inspecting the exact geometry. In this subsection, three techniques are presented.

The cross test

A simple technique to determine a hit using MBRs is the *cross test*: if the intersecting MBRs form a cross, the objects must intersect. The assumption that the area of a spatial object is contiguous is essential for the cross test. Figure 3 shows an example. Unfortunately, for real test data this test is very seldom successful. In our experiments, the cross test identifies only between 0.27% and 0.78% of the hits. Nevertheless, it is worth to execute the cross test because it needs only a few floating-point operations and no additional parameters.

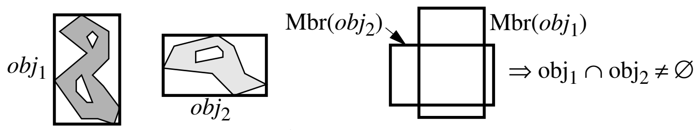


Fig. 3. Cross test

The false-area test

For two intersecting polygonal objects obj_1 and obj_2, the following property holds ($fa_{Appr}(obj)$ denotes the false area of the approximation $Appr(obj)$):

$$Appr(obj_1) \cap Appr(obj_2) > fa_{Appr}(obj_1) + fa_{Appr}(obj_2) \Rightarrow obj_1 \cap obj_2 \neq \emptyset$$

To say it in words, if the area of the intersection of the approximations is larger than the sum of the false areas of the objects, it follows that the objects intersect. This property can be exploited for processing spatial joins if the false area is stored additionally to the approximation for each object. This requires only one additional parameter. Now, the most interesting question is how many hits can be identified by the false-area test. Table 3 gives the percentage of identified hits in our experiments for various approximations.

MBR	RMBR	4-C	5-C	CH
0.05	0.5	2.5	6.2	10.1

Tab. 3. Percentage of hits identified by the false-area test (average on all test series)

Due to its bad approximation quality, the false-area test does not pay off for the MBR: almost no hit can be identified. Performing the test with the 5-corner about 6% of the hits are identified. These results motivate to look for a test which identifies more hits than the false-area test. For such a test, we allow a higher number of parameters.

Progressive approximations

In addition to conservative approximations which were discussed before, we will now consider another type of approximations. For identifying hits *progressive approximations* are adequate. A polygonal object is progressively approximated if the point set of the approximation is a subset of the point set of the object. When the low-cost test for progressive approximations is successful, we obtain a definite answer; only a failed operation triggers the costlier operation on the exact object description. For the intersection join this implies that if two progressive approximations intersect, it follows that the objects intersect.

It may be intuitively clear that it is more expensive to compute progressive approximations than conservative ones. This holds especially true if a maximum enclosed approximation is computed. In the following, we investigate three progressive approximations.

Enclosed circle

The simplest progressive approximation is the maximum *enclosed circle (EC)*. It can be computed for a simple polygon by using the Voronoi diagram of its edges (for the computation of such a diagram see e.g. [For 87]). One of the nodes of the diagram is the center of the maximum EC. The computation is illustrated in figure 4.

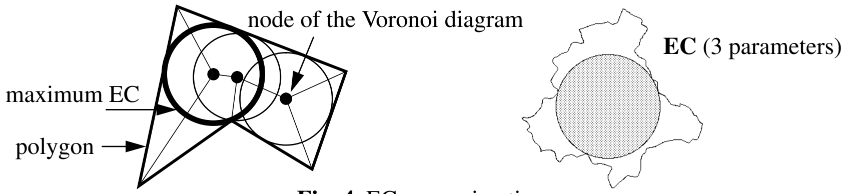


Fig. 4. EC-approximation

Enclosed rectangle

An obvious alternative to the EC is the maximum *enclosed rectangle (ER)*. In order to simplify the computation of such a rectangle, we restricted the investigation to rectangles which fulfill the following two properties: 1. they intersect the longest enclosed horizontal connection *h* starting in a vertex of the polygon and 2. the x- and the y-coordinates of the rectangles are x- and y-coordinates of vertices of the polygon. The ER is computed as follows: Based on a trapezoid decomposition [AA 83], the rectangles are computed which are visible from *h*. Using these rectangles as starting points, enclosed rectangles are constructed. Finally, the largest enclosed rectangle is selected. This process is illustrated in figure 5.

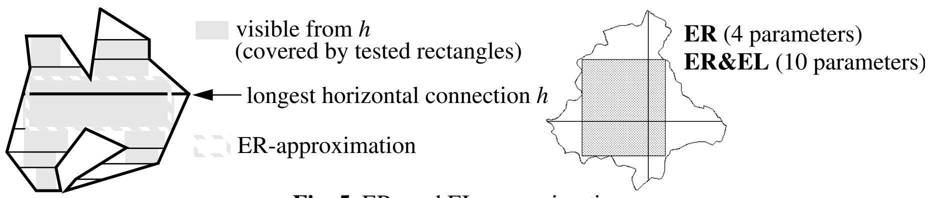


Fig. 5. ER- and EL-approximations

Enclosed line segments

The EC- and the ER-approximations are areas. However, it is also possible to use lines and line segments as progressive approximations. In this paper, we investigate a pair of maximum *enclosed rectilinear line segments (EL)*. Since non-area approximations cannot be used for query conditions which test whether another object is enclosed or not, in the following the EL-approximation will be considered in combination with the ER-approximation *(ER&EL)*. An example is given in figure 5.

Quality of filtering using progressive approximations

As mentioned before, progressive approximations allow us to identify hits. Table 4 shows the percentage of identified hits according to our experiments. The results in the column 'EC' demonstrate that almost 32% of the hits are identified by the EC-approximation without accessing the exact geometry. The ER-approximation even allows us to find around 35% of the hits. These percentages are considerably higher than those gained by the false-area test which identifies only about 6% using the 5-corner. The combination of the ER- and EL-approximation identifies up to 60% of the hits.

test series	EC	ER	ER&EL
Europe A	31.4	36.2	59.2
Europe B	31.8	35.3	57.7
BW A	31.6	34.3	59.0
BW B	32.6	33.6	59.7

Tab. 4. Percentage of identified hits using progressive approximations

Further experiments demonstrate that a combination of progressive approximations and the false-area test does not substantially improve the rate of identified hits. Therefore, it is not reasonable to use the false-area test in addition to progressive approximations.

3 The Impact of Additional Approximations on the MBR-Join

The last section has demonstrated the potentials induced by using conservative and progressive approximations. For a final assessment of those results, the impact of storing additional approximations on the MBR-join will be evaluated in this section. It is assumed that the approximations are stored in addition to the MBR (see [BKSS 94]) and that the MBR-join is performed by using *R*-trees* [BKSS 90].

Using additional conservative and progressive approximations, we need more parameters than in the traditional approach which is based on MBRs. The increased storage requirements decrease the capacity of a data page in the R*-tree, and hence worsen the performance of the MBR-join.

In order to investigate this effect, we joined two maps of real world objects each consisting of about 130,000 objects. As a result, 86,000 pairs of MBRs intersect. These data were already used in [BKS 93a]. The size of a page is 4 KB and the LRU-buffer holds 32 pages. The cost for a page access is 16 msec and the size of the entries varies between 46 and 206 Byte. Figure 6 depicts the results of this experiment.

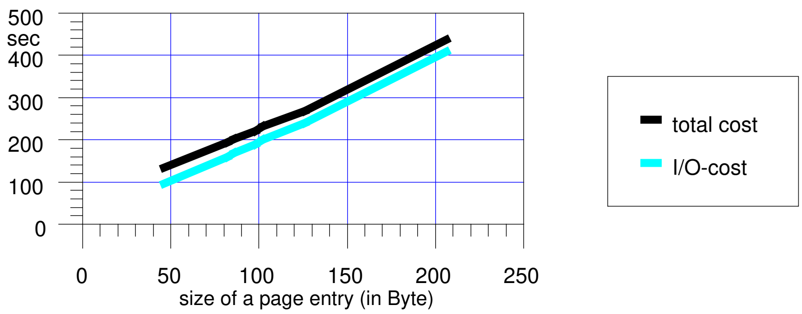


Fig. 6. The cost of the MBR-join depending on the size of the entries

The results show a linear dependency between the size of the page entries and the I/O-cost of the MBR-join. The CPU-cost of the MBR-join decreases with increasing entry sizes. Since we use the optimization techniques presented in [BKS 93a], the I/O-cost dominates the total cost for performing the MBR-join. As a consequence, the storage of larger page entries has a considerable impact on the cost of the MBR-join.

Therefore, we suggest to combine several physical pages (blocks) on secondary storage to one larger logical data page according to the enlargement of the page entries. The blocks of an enlarged data page are consecutively stored on the disk. The blocks of one enlarged data page cannot be individually read into main memory; reading the data page requires to access all its blocks. In this case, the transfer time for reading or writing a data page increases. However, the seek and latency time - which cause the major cost of a page access - will not be increased for reading or writing the page. Figure 7 demonstrates that the I/O-cost is only slightly affected by using additional approximations when larger data pages are used. For larger entries, the total execution time is considerably less than in figure 6. For the experiments, 9 msec are assumed for an average seek, 6 msec for the latency time and 1 msec for transferring 4 KB of data. These parameters are standard values for current disks [HS 94].

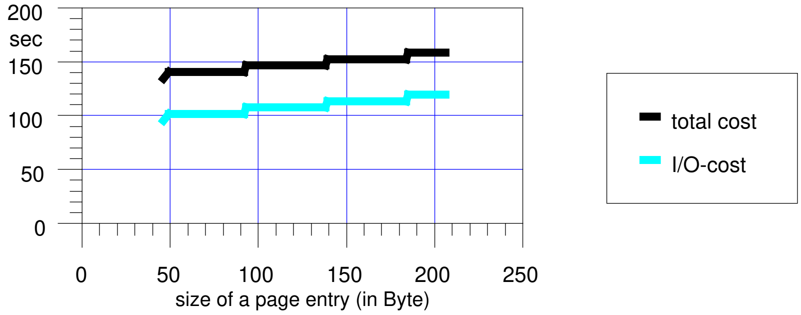


Fig. 7. The cost of the MBR-join depending on the size of the entries using enlarged data pages

4 The Impact of Additional Approximations on the Join

In this section, we investigate the impact of additional approximations with respect to the total execution time of an intersection join. Using the same test data as in the section before, we joined two maps. It is assumed that each description of an object stored in an R*-tree needs 16 Byte for the MBR and 32 Byte for additional information. One parameter of an approximation needs 4 Byte to be stored and the size of a directory page is 4 KB; the size of data pages depends on the size of the entries. The LRU-buffer holds 32 pages of the R*-tree and 1,600 pages for buffering the exact geometry of the objects. The average size of the description of the exact geometry of an object is 2.8 KB, i.e. on the average a polygon consists of 175 vertices.

The same seek, latency and transfer time as presented in the last section are assumed. The CPU-time spent in the approximation step for testing the intersection of two approximations is neglected. We assume that the exact geometry of two objects is tested for intersection by using a plane-sweep algorithm. Such a test needs 13.5 msec on an HP720-workstation.

We performed the joins with different combinations of approximations. Starting point of our investigation is the version which uses no additional approximations. The left column in figure 8 demonstrates that the cost for transferring the object into main memory (object access) and for testing the exact geometry for intersection (exact test) dominates the total execution time.

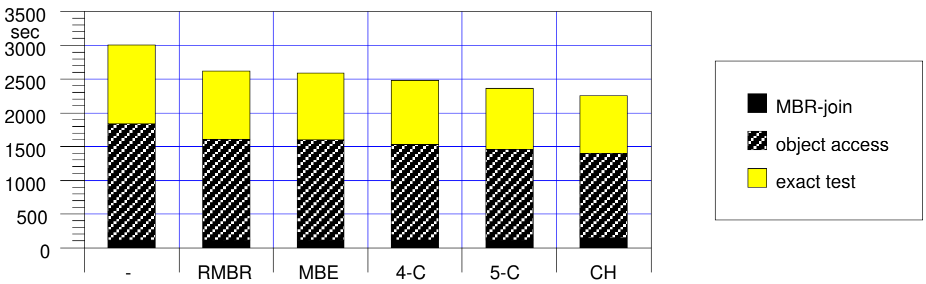


Fig. 8. Total execution time using different conservative approximations

The other columns in figure 8 depict the total execution time when different conservative approximations are used in addition to the MBR. Both, the object access and the exact intersection test are accelerated whereas the cost for the MBR-join is slightly increased. The results show the best performance for the convex hull. However, if other operations are also considered which do not take advantage of additional conservative approximations (e.g. the window query), the 5-corner is the best compromise between approximation quality and storage requirements.

The next diagram shows the performance of the intersection join using different progressive approximations. Now, only the cost of the exact geometry test is decreasing. The combination of the ER- and the EL-approximation has the best performance.

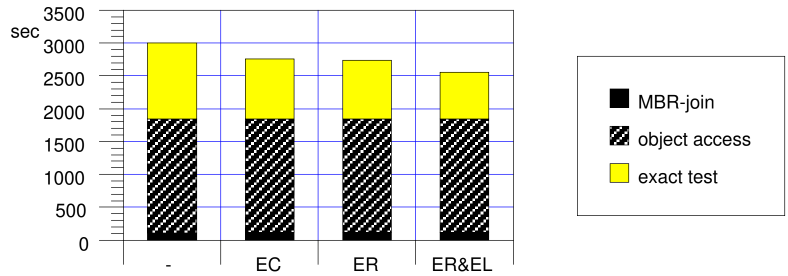


Fig. 9. Total execution time using different progressive approximations

Finally, we investigated the performance gains when the 5-corner and the combination of ER- and EL-approximation are used. The left diagram in figure 10 shows the execution time of the join which was investigated in the previous experiments. The total execution time is decreased by 30%. The right diagram shows the performance of an intersection join between similar maps. However, the selectivity is different: about 1.2 million pairs of MBRs intersect. In this case, the gain is almost 40%.

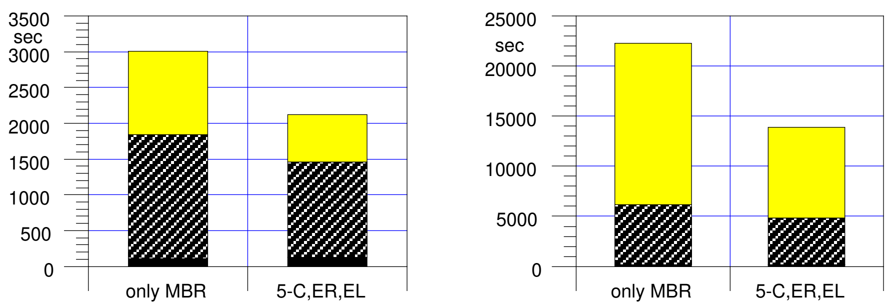


Fig. 10. Total execution time using conservative and progressive approximations

5 Conclusions

We investigated the problem of efficient spatial join processing using the intersection predicate. Our basic idea for efficient spatial join processing is to perform the join in several steps: In the first step, an MBR-join is performed. The approximation step works as a geometric filter and, finally, in the refinement step, the exact geometry is tested against the join predicate.

In this paper, we concentrate on the approximation step which detects a high fraction of false hits using conservative approximations in addition to the MBR. According to a detailed experimental investigation, we recommend to use the minimum bounding 5-corner of objects as an additional approximation. The 5-corner combines a high accuracy with a small storage overhead. In our experiments, about two thirds of the false hits in the candidate set computed by the MBR-join are identified by using the 5-corner. In order to detect hits in the candidate set (still without accessing the exact geometry), we propose to use simple progressive approximations, e.g. enclosed rectangles and enclosed line segments. A common intersection of two progressive approximations implies a common intersection of the corresponding objects. In our experiments, about 60% of the hits are detected by using the combination of enclosed rectangles and enclosed line segments as a progressive approximation.

The use of conservative and progressive approximations considerably accelerates the execution of an intersection join. Both, the performance of the object access and the performance of the exact intersection test, are improved.

In addition to the approximation step, we investigated the other steps for a multi-step processing of spatial joins: the MBR-join was examined in [BKS 93a], the test of the exact geometry of the objects and some other aspects of approximations in [BKSS 94], and the transfer of the objects from secondary storage into main memory in [BK 94].

So far, we assumed a conventional computer architecture. However, since the fast execution of spatial joins is extremely important, another task is to consider CPU- and I/O-parallelism in future work.

References

[AA 83] Asano Ta., Asano Te.: *'Minimum Partition of Polygonal Regions into Trapezoids'*, Proc. 24th IEEE Annual Symp. on Foundations of Computer Science, 1983, pp. 233-241.

[BG 90] Blankenagel G., Güting H.: *'Internal and External Algorithms for the Points-in-Regions Problem - the INSIDE Join of Geo-Relational Algebra'*, Algorithmica, 1990, pp. 251-276.

[BHF 93] Becker L., Hinrichs K., Finke U.: *'A New Algorithm for Computing Joins with Grid Files'*, Proc. 9th Int. Conf. on Data Engineering, Vienna, Austria, 1993, pp. 190-197.

[BK 94] Brinkhoff T., Kriegel H.-P.: *'The Impact of Global Clustering on Spatial Database Systems*', Proc. 20th Int. Conf. on Very Large Data Bases, Santiago de Chile, Sept. 1994.

[BKS 93a] Brinkhoff T., Kriegel H.-P., Seeger B.: *'Efficient Processing of Spatial Joins Using R-trees'*, Proc. ACM SIGMOD Int. Conf. on Management of Data, Washington, DC, 1993, pp. 237-246.

[BKS 93b] Brinkhoff T., Kriegel H.-P., Schneider R.: *'Comparison of Approximations of Complex Objects used for Approximation-based Query Processing in Spatial Database Systems'*, Proc. 9th Int. Conf. on Data Engineering, Vienna, Austria, 1993, pp. 40-49.

[BKSS 90] Beckmann N., Kriegel H.-P., Schneider R., Seeger B.: *'The R*-tree: An Efficient and Robust Access Method for Points and Rectangles'*, Proc. ACM SIGMOD Int. Conf. on Management of Data, Atlantic City, NJ, 1990, pp. 322-331.

[BKSS 94] Brinkhoff T., Kriegel H.-P., Schneider R., Seeger B.: *'Multi-Step Processing of Spatial Joins'*, Proc. ACM SIGMOD Int. Conf. on Management of Data, Minneapolis, MN, 1994, pp. 209-220.

[For 87] Fortune S. J.: *'A Sweepline Algorithm for Voronoi Diagrams'*, Algorithmica, Vol. 2, 1987.

[Gün 93] Günther, O.: *'Efficient Computation of Spatial Joins'*, Proc. 9th Int. Conf. on Data Engineering, Vienna, Austria, 1993, pp. 50-59.

[HS 94] Hilgefort U., Schneider R.: *'Rundablagen: Leistungsschau 47 neuer Festplatten'*, c't, No. 3, 1994, pp. 102-111.

[LR 94] Lo M.-L., Ravishankar C.V.: *'Spatial Joins Using Seeded Trees'*, Proc. ACM SIGMOD Int. Conf. on Management of Data, Minneapolis, MN, 1994, pp. 209-220.

[OM 88] Orenstein J.A., Manola F.A.: *'PROBE Spatial Data Modeling and Query Processing in an Image Database Application'*, IEEE Trans. on Software Engineering, Vol. 14, No. 5, 1988, pp. 611-629.

[Ore 89] Orenstein J. A.: *'Redundancy in Spatial Databases'*, Proc. ACM SIGMOD Int. Conf. on Management of Data, Portland, OR, 1989, pp. 294-305.

[Sam 90] Samet H.: *'Applications of Spatial Data Structures',* Addison-Wesley, 1990.

test series	# objects per map	# points per polygon	# intersecting MBRs	# hits	# false hits
Europe A / B	810	84	1858 / 4816	1273 / 3203	585 / 1613
BW A / B	374	527	2253 / 2562	1504 / 1684	749 / 878

Tab. 5. Test data

test series	# objects per map	# points per polygon	# intersecting MBRs	# hits	# false hits
Europe A	810	84	1858	1273	585
Europe B	810	84	4816	3203	1613
BW A	374	527	2253	1504	749
BW B	374	527	2562	1684	878

Tab. 6. Test data