On Local Heuristics to Speed Up Polygon-Polygon Intersection Tests

Wael M. Badawy
Center for Advanced Computer Studies
University of Southwestern Louisiana
Lafayette, LA 70504
wmb@cacs.usl.edu

Walid G. Aref
Department of Computer Sciences
Purdue University
West Lafayette, IN 47907
aref@cs.purdue.edu

ABSTRACT

The polygon-polygon intersection operation is CPU-intensive. Many data structures look into decomposing the polygons into multiple yet simple pieces to speed up the polygon-polygon intersection operation. This paper addresses local heuristics that can be adopted in these data structures by using local information about the simple polygon pieces to decide upon polygon-polygon intersections without having to perform this costly operation. The significance and effectiveness of each of the heuristics is studied. The paper also shows how these heuristics can be put together to perform a polygon join operation. Experiments are given to demonstrate the savings both in CPU and in I/O that result from these local heuristics.

Keywords

Spatial databases, polygon-polygon intersection, query processing

1. INTRODUCTION

We are given two polygons in vector format, and we are interested in finding whether these two polygons intersect. There are several good algorithms for detecting polygon-polygon intersection. What we are interested in is to see if we can still detect the intersection of two polygons when we have access only to parts of the polygon and not all of it.

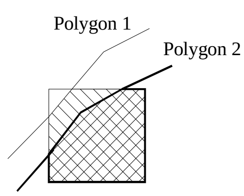

Figure 1: The intersection of the two polygons may be detected by only intersecting the polygon segments that lie in the square box.

As an illustration, consider the two polygon segments of Figure 1 Assume further that we only have access to the portions of the two polygons that lie inside the square box shown in the Figure. The rest of the polygon lies outside the box and we do not have access to the polygon segments that lie outside the box. The two shaded areas correspond to the inside regions of the two polygons. By storing the appropriate information with each polygon segment, we can possibly detect that the two polygons intersect without having to investigate the remaining portions of the two polygons.

The motivation behind this approach is that there exist many spatial data structures that store a polygon by subdividing the polygon into multiple yet simple pieces. It would be beneficial to be able to detect the polygon-polygon intersection given only partial information about one or more of the polygon pieces, without the need for retrieving the entire polygon. This paper presents useful heuristics that can be adopted to help detect the intersection of two polygons, or possibly all the intersecting pairs in two collections of polygons, using partial information about the polygons. As demonstrated by the experiments presented in the paper, this results in significant savings in CPU as well as I/O costs.

The rest of the paper proceeds as follows. Section 2 discusses the polygon representation that we assume in the paper. Section 3 presents the heuristics that we propose to avoid polygon-polygon intersection operations as much as possible, given the polygon representation in Section 2. Section 3 also demonstrates how this work can be extended to develop a polygon join algorithm in a database context. Section 4 gives experimental results that show the effectiveness of applying these heuristics. Section 5 contains concluding remarks and our plan for future research.

2. REPRESENTATION

There are many ways of decomposing a polygon into multiple yet simple pieces. For example, a polygon may be represented by a collection of triangles, rectangles, square blocks, trapezoids, convex polygons, etc. Our focus is on vector representation of a polygon.

In order to demonstrate our ideas, in this paper, we assume that a polygon is embedded into a coarse uniform grid (refer to Figure 2), where the space is split into equal-size grid cells. However, several variations of the data structure could be used. The reader is referred to [5][10][11] for extensive coverage of other spatial indexing techniques. The heuristics presented in this paper can be extended to other spatial data structures [12].

When embedded in a uniform grid, the vector representation of a polygon gets decomposed into a set polygon segments, each lies entirely in one of the grid cells. Moreover, there are grid cells that are entirely inside the polygon (e.g., grid cell 9 in Figure 2), and there are grid cells that are entirely outside the polygon (e.g., grid cells 0, 13, and 15 in Figure 2).

Each polygon, say P, has an identifier, say *pid*. However, instead of storing the polygon identifier of the polygon that intersects the grid cells, we store in a grid cell some summary information about the segment (part) of the polygon that overlaps the grid cell. This summary information help us in applying heuristics that let us detect polygon-polygon intersections without actually performing the polygon-polygon intersection operations.

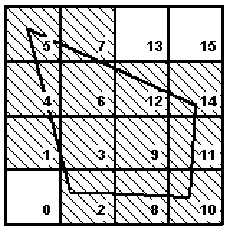

Figure 2: A polygon embedded in a grid.

Given a polygon P, we define a direction for each edge, say e, in P, so that each point, say p, inside P is to the right side of e. This is equivalent to assuming that each polygon has its edges pointing in the clockwise direction (refer to Figure 3). In this case, we say that e has a positive direction with respect to p.

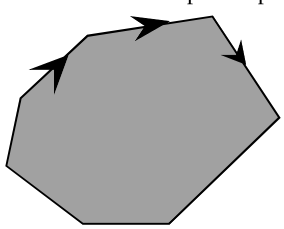

Figure 3: The polygon edge directions (clockwise direction)

We distinguish among three types of relationships that may occur between a grid cell and a polygon segment:

1. A polygon segment has one or more edges passing through a grid cell. In this case, the polygon segment is termed an edge segment (EP).

2. A polygon segment fully covers (contains) a grid cell. In this case, the polygon segment is termed a covering segment (CP).

3. A polygon segment has no relationship with a grid cell.

Table 1 illustrates the relationships among the grid cells and the polygons that appear in Figure 2.

Grid cells having EP polygon segments	1, 2, 3, 4, 5, 6, 7, 8, 10, 11, 12, 14.
Grid cells having CP polygon segments	9.
Grid cells having no polygons passing through them	0, 13, 15.

Table 1: Summary of the relation between polygon and grid cell of Figure 2

For an edge segment the border of the grid cell is used to close the corresponding polygon segment, and form a region (see Figure 4). In the Figure, the parts XW, WZ, and ZY of the grid cell border close the polygon segment XY.

The active control point of a polygon segment within a grid cell is defined as the starting point in which the polygon edge begins to intersect the grid cell border. For example, in Figure 4, point Y is the active control point.

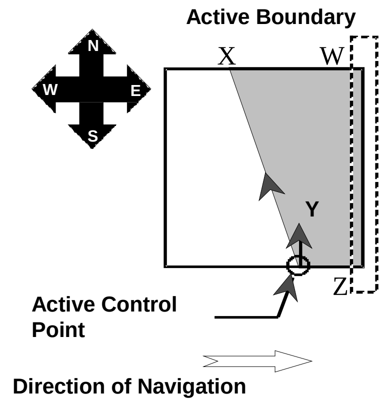

Figure 4: The grid cell layout

The direction d of a polygon segment is defined as a vector (X, Y). We use the direction of the Y component of the polygon segment at the active control point as the segment direction. For example, in Figure 4, the direction of polygon segment XY is the south-north direction. Furthermore, a polygon segment is positive (i.e. d=1) if the direction of the polygon segment is the same as the Y direction of the active control point.

We refer to the eastern border of a grid cell as its active border [10]. The active border holds some information after the currently visited grid cell and this information gets carried over to the next grid in the eastern direction.

More specifically, we store the following information in a grid cell for every EP polygon segment that intersects the cell: SEG(pid, a, b, d), where pid is an identifier of the polygon that owns this segment, a and b are two characteristics that describe the behavior of the EP segment, and d is the segment direction. The values of a, b, and d are assigned for each EP segment as defined below.

The attribute a can have one of the following values:

a = 1 if the polygon segment is formed by a chain of one or more polygon edges, where none of the edges intersects the active border (the eastern border) of the grid cell.

a = 2 if the polygon segment is formed by a chain of one or more polygon edges, where at least one of those edges intersect the active border of the grid cell.

The attribute b can have one of the following values:

b = 1 if the polygon segment is formed by a chain of one or more polygon edges, where the edges intersect two opposite grid cell boundaries.

b = 2 if the polygon segment is formed by a chain of one or more polygon edges, where the edges intersect two neighbor grid cell boundaries. There are four possible polygon segment types depending on which two neighboring sides are intersected by the polygon segment.

b = 3 if the polygon segment is formed by a chain of one or more polygon edges, where the edges intersect one grid cell border. There are four possible cases; one for each border of the grid cell.

b = 4 if the polygon segment is formed by a chain of one or more polygon edges, where the edges intersect three or more grid cell boundaries.

b = 5 when none of the above cases apply.

As mentioned above, each polygon is split into two types of polygon segments: EP and CP polygon segments. Inside the cells of the uniform grid, the EP polygon segments are stored explicitly (in the form of the tuple SEG(pid, a, b, d)) and not the CP polygon segments. This saves in the size of the index as the detection of the existence of a CP polygon segment in a cell can be done dynamically. An auxiliary dynamic data structure is used for this purpose and is termed the propagation list.

Two cases arise during the inspection of the polygon edges with respect to a certain grid cell:

1 Some edges of polygon P pass through the grid cell g. Then, the polygon identifier of p is explicitly stored in an EP segment in grid cell g.

2. The polygon P does not have a polygon identifier in a grid cell g. Then P either does not intersect g or fully contains (covers) g.

We maintain a propagation list that contains the polygon identifiers of the polygons that are expected to fully cover a grid cell. The propagation list is passed from the grid cell g1 to the grid cell to its east, i.e., to g2, after the insertion or deletion of some items in the list.

The propagation list data structure is most beneficial in the case of intersecting two collections of polygons together, e.g., as in the case of the polygon join operation. We make use of it in Section 3.2.

In the following section, we show how the propagation list for CP segments as well as the tuples stored for each EP segment aid in detecting polygon-polygon intersections. This is performed by applying a set of heuristics. When applicable, these heuristics help replace the costly polygon-polygon intersection operation by simple tests over the values of a, b, and d for each polygon segment.

3. HEURISTICS

In this section, we present two groups of heuristics. The first group of heuristics applies when we have only two polygons as input, and we want to detect if they possibly intersect. In contrast, the second group of heuristics applies when we have as input two sets of polygons and we want to detect all the polygon pairs from the two sets that intersect with each other.

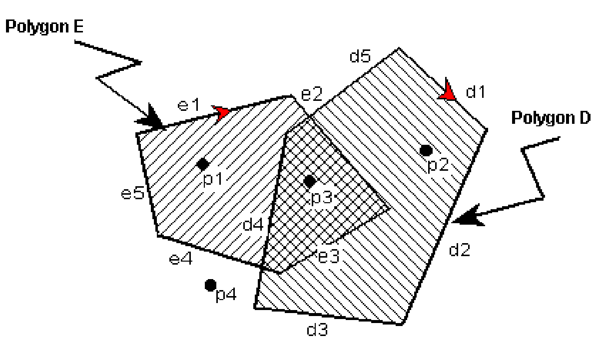

Figure 5: The directed polygons Intersection

Figure 5 illustrates the two polygons D and E, where d1, d2, d3, d4, and d5 are edges of D, and e1, e2, e3, e4, and e5 are edges of E. The directions of the edges are shown in Figure 5. Consider the four points *p1, p2, p3* and *p4* as shown in the Figure. Point *p1, is* located in polygon E but not D, because d4 is in counter-clockwise direction with respect to *p1*. Point *p2, is* located in polygon D but not E, because e2 is in counter-clockwise direction with respect to *p2*. Point *p3, is* located in polygon D and E. Finally, point *p4* is located outside either polygon because d4 and e4 are in counter clockwise direction with respect to *p4*.

3.1 Heuristics for Detecting Whether Two Polygons Intersect

Several approaches to polygon-polygon intersection are already present in the literature, e.g., [2], [7], [9]. We are interested in developing some heuristics that help avoid performing such an operation.

Let P1 and P2 be two polygons that are embedded into grid cells and are partitioned into polygon segments. Furthermore, Let S1i and S2i be two polygon-segments of polygons P1 and P2, respectively, that overlap grid cell I. The heuristics presented in this section help in detecting whether the polygons P1 and P2 intersect using local information about the polygons corresponding segments (S1 and S2).

Heuristic 1:

Polygons P1 and P2 intersect if there exist two polygon segments S1i (P1, a1, b1, d1) and S2i (P2, a2, b2, d2) in a grid cell i, where a1 = 1 and d1= d2.

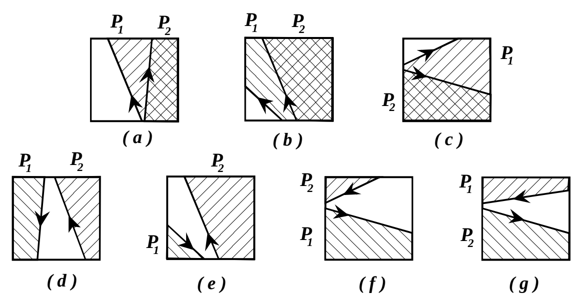

Figure 6: Example applications of Heuristic 1: (a)-(c) Polygons P1 and P2 intersect each other, (d)-(g) the polygons do not intersect each other.

Refer to Figure 6. In Figure 6a-6c, polygon P1 has its a1 = 1. This means that P1 does not intersect the eastern border of the grid cell. Moreover, since the two polygon segments for P1 and P2 have the same edge directions (since d1= d2), then we know that both polygons intersect each other. Counter examples are given in Figures 6d-f. For example, Figure 6d--6e show polygon segments that have opposite edge directions inside the grid cell and hence do not intersect. Figures 6f--6g show polygon segments that both intersect the eastern border (i.e., a1 □ 1) and hence the heuristic is not applicable, as we cannot always guarantee the intersection of the two polygons.

Heuristic 2:

Two polygons P1 and P2 intersect if there exist two polygon segments S1i (P1, a1, b1, d1) and S2i (P2, a2, b2, d2) in a grid cell i, where a1 = 1, a2 =2, b1= 1, and b2 = 1.

We use Figure 7 for illustration. In the Figure, the two polygon segments are guaranteed to always intersect because of the conjunction of the following two conditions: (1) b1= 1 and b2 = 1 mean that each of the two segments intersect two opposite boundaries, e.g., the east and west boundaries or the north and south boundaries. (2) a1 = 1 and a2 =2 mean that one of the two polygon segments intersect the eastern border (the active border) while the other does not. The two conditions combined guarantee that both polygon segments have to intersect some where inside the grid cell.

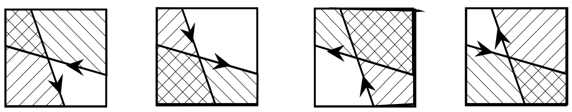

Figure 7: Examples of the applicability of Heuristic 2.

Heuristic 3:

If S1(P1, a1, b1, d1) is a polygon segment where a1= 1, b1=1, and d1=1, then most probably P1 has a CP segment type in the next grid cell (the eastern neighbor of the current cell).

The explanation of this Heuristic is as follows. Having a1= 1, b1=1, and d1=1 (refer to Figure 8) mean that the end points of the polygon segment inside the grid cell intersect the northern and southern borders of the grid cell. Therefore, two cases may happen. (1) The polygon covers the next grid cell (the one to the right of the current grid cell), as in Figure 8a, and hence forms a segment of type CP. (2) The polygon has another edge that intersects the next grid cell and hence does not entirely cover the next grid cell, as in Figure 8b.

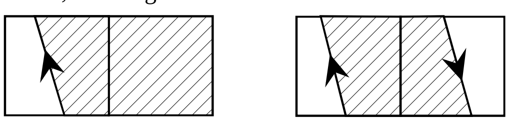

Figure 8: (a) The polygon entirely covers the next grid cell to the right, (b) the polygon does not cover the next grid cell as when of its other edges passes through the grid cell.

The significance of this heuristic is that it helps detect when a polygon covers an entire grid cell. In this case, that polygon would intersect all the other polygons that overlap with the grid cell. More specifically, when the precondition of the heuristic is met, i.e., when a1= 1, b1=1, and d1=1 for a segment, say s1, that belongs to a polygon, say P1, we insert the polygon identifier of P1 into the propagation list. When visiting the next grid cell, say c, we check if there exists in c any polygon segment that refers to P1. If such a polygon segment does not exist, then we know that P1 entirely covers the grid cell c. In this case, we can deduce that P1 intersects all the polygons that overlap with c (refer to Heuristic 4). If there exists any polygon segment in c that is part of P1, then we remove P1 from the propagation list. The reason is that we cannot deduce in this case that P1 covers c. The usefulness of this heuristic is more significant when we deal with two collections of polygons and not just two polygons.

3.2 Heuristics for Detecting All Pairs of Polygon Intersections in Two Sets of Polygons

The heuristics in this section apply when we are intersecting two sets of polygons and are interested to find all the polygon pairs that intersect with each other. In the database literature, this operation is often termed a polygon join operation.

More formally, let S1 and S2 be two sets of polygons. The result of joining S1 and S2 is a list of all polygon pairs (Pi , Gj), where Pi and Gj are two distinct polygons, that belong to S1 and S2, respectively, such that, for all i, j, Pi intersects Gj.

There are many existing vector-based polygon join algorithms, e.g., see [1],[3],[6] and [8]. In this paper we are interested in developing heuristics that may work in conjunction with these algorithms. The target of the heuristics is to further speed up the polygon join operation.

All the heuristics, presented in Section 3.1, for the two-polygon case, also apply in the case of detecting intersections in sets of polygons. Additional heuristics apply only for the latter case. These are listed below.

Heuristic 4:

If S1 (P1, a1, b1, d1) is a polygon segment in grid cell i, then P1 intersects all the polygons that have CP segments in the propagation list at the time of visiting grid cell i.

As stated in Section 31, this heuristic works along with Heuristic 3 to detect whether a polygon covers an entire grid cell. In this case, we can report the intersection of the polygon with all the polygons that overlap with the grid cell.

Heuristic 5:

Assume that there exist j polygon segments Sji(pj, aj, bj, dj), in grid cell gi. Sji(pj, aj, bj, dj) is grouped into multiple groups Gk(p, a, b, d) based on the values of a, b, and d. Any optimization rule from the ones listed above that applies to a polygon segment S in G applies as well to all the other segments of G.

Figure 9 shows polygon segments S1, S2, and S3. These segments can be divided into two groups: G1 that contains the polygon segments S1 and S2, and G2 that contains S3. In Figure 9, S1 is selected to represent G1, and S2 represents G2. Notice that there are no constrains in selecting any segment to be the representative segment as long as it has the same attribute values as of the other segments in the same group. Notice further that deciding whether two segment groups intersect or not depends only on the attributes values of the group representative and not on the segments' shapes.

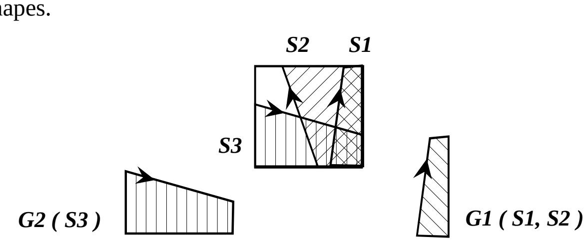

Figure 9: Grouping of polygon segments

4. EXPERIMENTAL SETUP AND RESULTS

In order to study the effectiveness of the proposed heuristics, we perform the following experiments. We get two sets of polygons. Each set is composed of a collection of polygons that cover a certain region in space. We assume that each set of polygons is embedded in a uniform grid. We store in each grid cell only the segment (part) of the polygon that overlaps the grid cell, as described in Section 2. We try to detect all the intersecting polygon pairs. We repeat this experiment twice: once without using the heuristics and once when using the heuristics.

When not using the heuristics, we apply the polygon merge algorithm, listed below. The core of this algorithm is the polygon-polygon intersection test.

Algorithm Polygon-Merge

1) Traverse the two uniform grids, say G and H, simultaneously and retrieve the grid cells that are spatially registered with each other (i.e., the ones from both grids that occupy the same space).

2) For each pair of spatially registered grid cells, say gi and hj, such that gi and hj belong to G and H, respectively:

3) For all polygon identifiers pidg in gi and pidh in hi

4) Retrieve the polygons P(pidg) and P(pidh) whose identifiers are pidg and pidh.

5) Perform a polygon-polygon intersection test between P(pidg) and P(pidh).

6) If they intersect, then report P(pidg), P(pidh) as an intersecting output pair

7) End.

In order to test the heuristics, we modify Algorithm Polygon-Merge so that it tries to apply any of the heuristics before performing the polygon-polygon intersection test.

Both real and synthetic data sets are used in the experiments. The real data sets used in the experiments consist of maps of road networks of counties and cities in the USA obtained from the US Bureau of Census Tiger/Line database of roads and other geographic features in the USA [4]. A line consisting of consecutive straight-line segments represents each road. Seven real data sets are used. Their names and characteristics are given in Table 2. Polygonal objects are constructed from these line segments. The process for constructing the polygonal objects is skipped here for brevity. In order to increase the population of the database, the data sets are replicated in some fashion, so that we were able to increase the size of the database to around 1,000,000 polygonal segments. In addition, various synthetic data sets are generated using the normal distribution and the pivot space distribution (i.e., the polygonal objects are clustered around certain pivot point in the space).

State	Data collection	No. polygons
Prince George, MD	PRINCE	28884
Baltimore, MD	BALTIMORE	19896
Washington, DC	DC	11165
Franklin, VA	FRANKLIN	837
Bedford, VA	BEDFORD	790
Williamsburg, VA	WILLIAMS	681
Falls Church, VA	F.CHURCH	500

Table 2: The real data sets used for the experiments.

Figure 10 gives the total number of polygon-polygon intersection tests performed by algorithm Polygon-Merge algorithm (denoted by Poly-Poly in the Figure), the number of polygon-polygon tests that are avoided as we apply the heuristics. Notice that as the number of polygons in the database increases, the portion of the polygon pairs that are detected using the heuristics. This portion is around 45% of the total tested pairs. From this we can observe the usefulness and significance of using these heuristics.

Notice that the usage of heuristics does not exclude the necessity, in some cases, to perform a polygon-polygon intersection test. For example in Figure 11, the heuristics do not help on avoiding the execution of a polygon-polygon intersection test.

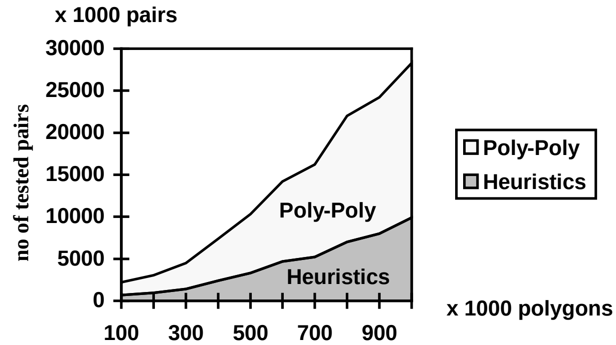

Figure 10: The average number of tested polygon pairs (real data sets)

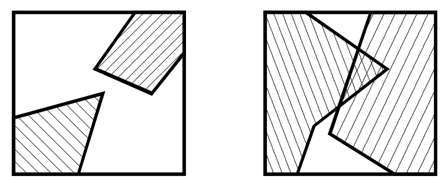

Figure 11: Two example cases where the heuristics do not help in avoiding a polygon-polygon intersection operation.

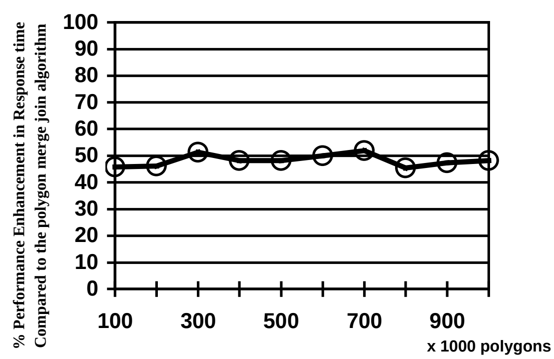

Figure 12: The percentage enhancement in response time when using the heuristics over the polygon merge join algorithm with polygon-polygon intersection operation (real data sets).

Figure 12 shows the percentage enhancement in response time when using the heuristics over the polygon-merge join algorithm with the polygon-polygon intersection operation. The graph shows around 50% improvement in response time. This enhancement in response time is due to two factors: (1) reduction in the CPU cost and (2) reduction in the I/O cost. The reduction in CPU cost is due to the reduction in the number of polygon-polygon intersection tests that are performed. On the other hand, the reduction in I/O cost is because with every application of a heuristic, this means that we were able to detect an a polygon-polygon intersection using the local information inside the grid cell. This saves in I/O time since we do not have to retrieve the polygon data from disk in order to test for intersection.

Figure 13 gives the percentage improvement in the response time of the join algorithm while varying the grid cell size. The size of the grid cell influences the grid cell capacity, which indicates the density of the polygons per gird cell. The figure shows that the increase in the polygon density in the grid cells reduces the response time. On the other hand, the algorithm that uses the heuristics benefits from the high density of the polygonal objects in the gird cell (i.e., the increase in the number of polygon pairs), since the saving in the cost of the intersection test enhances the response time.

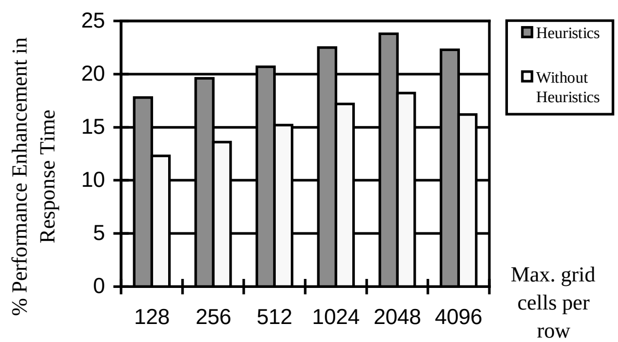

Figure 13: The effect of the grid cell size on the performance of the join algorithms

5. CONCLUDING REMARKS

We have demonstrated the usefulness of the heuristics in reducing the number of polygon-polygon intersection tests (by around 45%) as well as in reducing the overall response time of the polygon-merge join algorithm (by around 50%). The reduction in the response time is due to the reduction in both the CPU and I/O costs, as explained in Section 5.

Our future work includes investigating the possibility of extending some of the existing spatial join algorithms with the heuristics proposed in this paper in an attempt to enhance the performance of these algorithms.

6. REFERENCES

[1] Lars Arge, Octavian Procopiuc, Sridhar Ramaswamy, Torsten Suel, Jeffrey Scott Vitter, Scalable Sweeping-Based Spatial Join, VLDB 98.

[2] Walid G. Aref and Hanan Samet. The Spatial Filter Revisited. Proceedings of the 6th Intl. Symposium on Spatial Databases Handling, UK, 1994, pp.190-208.

[3] T. Brinkhoff, H.P. Kriegel, and B. Seeger. Efficient processing of spatial joins using R-trees. Proceedings of the ACM-SIGMOD International Conference on Management of Data, Washington, D.C., pages 237-246, May 1993.

[4] Bureau of the Census: 1990 Technical Documentation. Tiger/Line Census files. Technical report, US Bureau of Census, Washington, DC, 1989.

[5] Volker Gaede and Oliver Gunther, Multidimensional Access Methods, ACM Computing Surveys, June 1998.

[6] Erik G. Hoel, Hanan Samet, Benchmarking Spatial Join Operations with Spatial Output. 606-618, VLDB 1995.

[7] J. O'Rourke, C.-B Chien, T.Olson and D.Naddor, A new linear algorithm for intersecting convex polygons. Computer Graphics and Image Processing Proc. 19, 1982, 384-391.

[8] Jignesh Patel and David DeWitt: Partition-Based Spatial-Merge Join, Proceedings of the ACM-SIGMOD International Conference on Management of Data, 1996.

[9] Preparata, Franco P. and Shamos, Micheal Ian, Computational Geometry: An Introduction, Springer-Verlag 1985.

[10] H. Samet, Applications of Spatial Data Structures, Reading, MA: Addison Wesley, 1990.

[11] H. Samet. Spatial Data Structures. In Won Kim, editor, Modern Database Systems, Addison-Wesley, Reading, Massachusetts, 1992, Ch. 18, pp. 361-385.

[12] Wael Badawy, A New Spatial Join Algorithm For Large Polygonal Databases, Master's Thesis, Alexandria University, Egypt, 1997.