Summarizing Level-Two Topological Relations in Large Spatial Datasets

XUEMIN LIN
NICTA & University of New South Wales, Australia
QING LIU
NICTA & University of New South Wales, Australia
HONGJUN LU
Hong Kong University of Science and Technology, Hong Kong
YIDONG YUAN
NICTA & University of New South Wales, Australia
XIAOFANG ZHOU
University of Queensland, Australia

Summarizing topological relations is fundamental to many spatial applications including spatial query optimization. In this paper, we present several novel techniques to effectively construct cell density based spatial histograms for range (window) summarizations restricted to the four most important level-two topological relations: contains, contained, overlap, and disjoint. We first present a novel framework to construct a multiscale Euler histogram in 2D space with the guarantee of the exact summarization results for aligned windows in constant time. To minimize the storage space in such a multiscale Euler histogram, an approximate algorithm with the approximate ratio 19/12 is presented, while the problem is shown NP-hard generally. To conform to a limited storage space where a multiscale histogram may be allowed to have only k Euler histograms, an effective algorithm is presented to construct multiscale histograms to achieve high accuracy in approximately summarizing aligned windows. Then, we present a new approximate algorithm to query an Euler histogram that cannot guarantee the exact answers; it runs in constant time. We also investigate the problem of non-aligned windows and the problem of effectively partitioning the data space to support non-aligned window queries. Finally, we extend our techniques to 3D space. Our extensive experiments against both synthetic and real world datasets demonstrate that the approximate multiscale histogram techniques may improve the accuracy of the existing techniques by several orders of magnitude while retaining the cost efficiency, and the exact multiscale histogram technique requires only a storage space linearly proportional to the number of cells for many popular real datasets.

Categories and Subject Descriptors: H.2.4 [**Database Management**]: Systems—*Query Processing*; H.3.3 [**Information Storage and Retrieval**]: Information Search and Retrieval—*search process*

General Terms: Histograms, Spatial Query Processing and Optimization

1. INTRODUCTION

Research in spatial database management systems (SDBMS) has a great impact on many applications, including geographic information systems, digital libraries, robotics, image processing, CAD and VLSI. In the last 20 years, indexing and query

A preliminary version was published in [Lin et al. 2003]. The work was partially supported by an ARC Discovery Grant DP0345710.

processing in SDBMS have drawn a great deal of attention from research society. A number of techniques have been developed [Gaede and Günther 1998; Lo and Ravishankar 1996; Mamoulis and Papadias 1999; Patel and DeWitt 1996]. With the recent availability of massive on-line spatial data, there are strong demands [Greene et al. 1999; Sun et al. 2002a; Szalay et al. 2000] for developing efficient techniques to browse large datasets for summarising spatial characteristics, so that users can quickly identify relevant data among enormous available data resources. It becomes extremely important in large digital libraries/archives to support *interactive* queries by *query preview* [Beigel and Tanin 1998; Greene et al. 1999]. Summarizing spatial datasets also plays an important role in spatial query processing optimization by providing selectivity estimation [Aboulnaga and Naughton 2000; Acharya et al. 1999; Jin et al. 2000; Sun et al. 2002b].

Various techniques [Garofalakis and Gehrke 2002; Gilbert et al. 2002] have been recently developed for effectively summarizing relational datasets. The most common techniques are *samples/sketches* [Garofalakis and Gehrke 2002; Lipton et al. 1990], *histograms* [Poosala 1997; Garofalakis and Gehrke 2002], and *wavelets* [Matias et al. 1998]. These paradigms have been extended to summarizing topological relations against spatial datasets.

In this paper, we investigate the problem of summarizing *rectangular* objects for range (window) queries. Several histogram based summarization techniques have recently been developed to provide selectivity estimation for spatial range queries. The existing techniques may be divided into two categories: 1) *data partition techniques*, and 2) *cell density.* The Min-skew algorithm in [Acharya et al. 1999] and the SQ-histogram technique in [Aboulnaga and Naughton 2000] belong to the first category, and propose to group "similar" objects together according to some mathematical models to form a bucket for estimating the number of *disjoint* objects and the number of *non-disjoint* objects in window queries.

Techniques based on cell density [Beigel and Tanin 1998; Jin et al. 2000] propose to divide the data space into a number of *disjoint* cells, and to record some kind of object density for each cell. To estimate the number of non-disjoint objects against a window, a cumulative density based approach (CD) was proposed in [Jin et al. 2000], while the *Euler* formula [Harary 1969] has been effectively used in [Beigel and Tanin 1998] for creating a cell density based histogram (called "Euler histogram").

The above techniques were developed for summarizing the *level-one topological* relations [Egenhofer and Herring 1994; Grigni et al. 1995; Sun et al. 2002a]: disjoint and non-disjoint. The problem of summarizing level-one topological relations has also been investigated for spatial temporal datasets; the techniques may be found in [Hadjieleftheriou et al. 2003; Tao et al. 2003; Tao et al. 2004]. In contrast, little has been done for summarizing *level-two* topological relations [Egenhofer and Herring 1994; Grigni et al. 1995; Sun et al. 2002a] though the level-two topological relations are equally important for spatial window queries.

To the best of our knowledge, [Sun et al. 2002a] is the only paper investigating the problem of summarizing the level-two topological relations. It applies the Euler histogram techniques in a novel way for this purpose and demonstrates that a cell density based histogram technique is very effective to approach such a problem.

In this paper, we study the problem of effectively summarizing the level-two topological relations using cell density based histogram techniques. In the context of spatial data, a *minimum bounding rectangle* (MBR) is usually used to approximately represent an arbitrary object for the purpose of data summarization. Therefore, our investigation is restricted to summarizing rectangular objects.

As with relational datasets, the accuracy of such summarization against *aligned windows*, which consist of whole cells only, is fundamental to the quality of a cell density based histogram. Moreover, a cell density based histogram is *practical* if its storage space is linearly proportional to the number of the disjoint cells to be used. The histogram techniques in [Beigel and Tanin 1998; Jin et al. 2000; Sun et al. 2002a] are practical, and guarantee the exact solutions for computing the number of level-one topological relations against aligned windows.

The histogram techniques in [Sun et al. 2002a] can also provide a highly accurate approximate solution for summarizing level-two topological relations. However, they rely on a few strong assumptions; the accuracy may greatly degrade if the spatial data do not conform to the assumptions. Motivated by this, in the paper we propose a novel multiscale paradigm by effective utilization of the object "scales" (to be precisely defined in section 3). It consists of two steps: 1) group together the objects with "similar" scales, and 2) for each group of objects, create an Euler histogram. The main contributions of the paper may be summarized below.

(1) For a given "resolution" (to be defined in section 2), we propose a novel framework to construct a multiscale histogram in a 2-dimensional space that is composed of multiple Euler histograms. Such a multiscale histogram can generate exact summarization results for aligned windows in constant time for level-two topological relations.

(2) We investigate the problem of storage space minimization in a multiscale histogram. We show that the problem is generally NP-hard, and an application of the *Duh-Furer semi-local* optimisation technique [Duh and Fürer 1997] can guarantee the approximate ratio 19/12.

(3) To conform to a limited storage space where only k Euler histograms are allowed, we present an effective algorithm to construct multiscale histograms with high accuracy. We also develop a novel approximate algorithm to query an Euler histogram that cannot guarantee exact answers; the algorithm runs in constant time.

(4) We present new techniques to deal with non-aligned windows and to effectively dividing the data space for a finer resolution. We extend our techniques from a 2-dimensional space to a 3-dimensional space.

We evaluate our new techniques by both synthetic and real world datasets. Our experiment results demonstrate that the approximate multiscale techniques may improve the accuracy of the existing techniques by several orders of magnitude while retaining the cost efficiency. The experiments also show that the exact multiscale histogram technique requires only a storage space linearly proportional to the number of disjoint cells for many popular real world datasets; that is, it is practical.

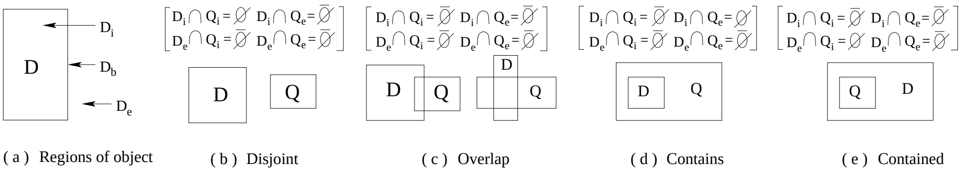
The rest of the paper is organized as follows. In section 2, we provide preliminaries and related work. In section 3, we present our first and second contributions of the paper - efficient histogram construction algorithms for generating exact summarization results against aligned windows while minimizing the storage space.

Section 4 presents the third contribution of the paper. Sections 5-7 present our fourth contribution. Section 8 presents a discussion on maintenance, generalization, and application of our techniques. Section 9 presents the experiment results. This is followed by conclusion and remarks.

2. PRELIMINARIES

In this paper, we study only *axis-aligned* rectangular objects, since different types of objects are usually represented by their minimum bounding rectangles (MBR) to approximate the spatial extents for data summarization. A set S of objects, in this paper, always means a set of axis-aligned rectangles in 2D space except section 7 where we address 3D space.

A binary topological relation between two objects, D and Q, is based upon the comparison [Egenhofer and Herring 1994] of D's *interior* D_i, *boundary* D_b, *exterior* D_e (see Figure 1(a)) with Q's interior, boundary, and exterior. It can be classified [Egenhofer and Herring 1994; Grigni et al. 1995; Sun et al. 2002a] into 8 high-resolution (level 3) topological relations according to the 9-intersection model, and can be also classified into the 5 medium-resolution (level 2) topological relations by omitting 3 less important relations with boundary meeting tests. In this paper, we focus only on the 4 medium-resolution topological relations (as depicted in Figures 1(b) - (e)) - disjoint (ds), overlap (ov), contains (cs), and contained (cd) by omitting the *equal* relation since equal relation may be approximately treated as contains or contained relation.


D_i
D_b
D_e
D
$\begin{bmatrix} D_i \cap Q_i = \emptyset & D_i \cap Q_e = \bar{\emptyset} \\ D_e \cap Q_i = \bar{\emptyset} & D_e \cap Q_e = \bar{\emptyset} \end{bmatrix}$
D
Q
$\begin{bmatrix} D_i \cap Q_i = \bar{\emptyset} & D_i \cap Q_e = \bar{\emptyset} \\ D_e \cap Q_i = \bar{\emptyset} & D_e \cap Q_e = \bar{\emptyset} \end{bmatrix}$
D
Q
D
Q
$\begin{bmatrix} D_i \cap Q_i = \bar{\emptyset} & D_i \cap Q_e = \emptyset \\ D_e \cap Q_i = \bar{\emptyset} & D_e \cap Q_e = \bar{\emptyset} \end{bmatrix}$
D
Q
$\begin{bmatrix} D_i \cap Q_i = \bar{\emptyset} & D_i \cap Q_e = \bar{\emptyset} \\ D_e \cap Q_i = \emptyset & D_e \cap Q_e = \bar{\emptyset} \end{bmatrix}$
Q
D

(a) Regions of object (b) Disjoint (c) Overlap (d) Contains (e) Contained

Fig. 1. 4 Topological Relations between Two Objects

Below, we give a brief overview of Euler histograms [Beigel and Tanin 1998], and the three algorithms (S-Euler, EulerApprox, and M-Euler) [Sun et al. 2002a] to query an Euler histogram. These techniques are closely related to our work in this paper. Then, we brief the motivation of this research.

2.1 Euler Histograms

To build an Euler histogram H for a set S of objects, a *resolution* is pre-given; that is, the axis-aligned MBR containing the whole S is first divided into $n_1 \times n_2$ disjoint cells, also called a *grid* of H or S. For instance, Figure 2(a) illustrates the 5×4 grid.

Note that in an $n_1 \times n_2$ grid, each node on the grid is called *grid point* and labelled by (i, j), lexicographically, with the integers i and j in the range of $1 \leq i \leq n_1 + 1$ and $1 \leq j \leq n_2 + 1$. There are n_1 cells spanning the grid horizontally and n_2 cells spanning the grid vertically; thus, n_1 is the *width* and n_2 is the *height* of the grid. The total number of cells, internal nodes, and internal edges is $(2n_1 - 1) \times (2n_2 - 1)$.

In an Euler histogram with a resolution $n_1 \times n_2$, $(2n_1 - 1) \times (2n_2 - 1)$ buckets

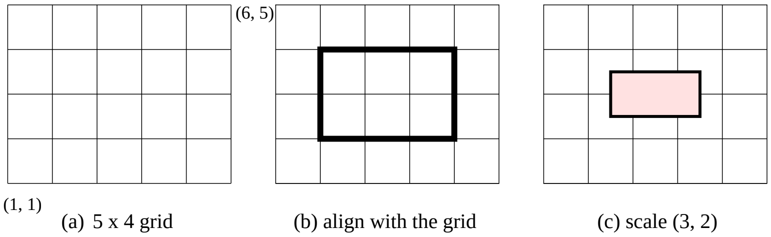


Fig. 2. Grid, Query Window, and Object Scale

are given where each cell is assigned a bucket to store an integer with initial value 0 as well as each internal edge and each internal node are allocated a bucket, such that [Beigel and Tanin 1998; Sun et al. 2002a]:

—The integer, corresponding to a cell in the grid, is increased by 1 if an object intersects the cell.

—The integer, corresponding to a node (grid point) in the grid, is increased by 1 if an object contains the node.

—The integer, corresponding to an edge in the grid, is decreased by 1 if an object crosses the edge.

Figure 3(a) gives an example of an Euler histogram. Note that in an Euler histogram H, we do not deal with the information that a boundary of an object *aligns with* the grid of H (see Figure 2(b) for example). This is because we can always "shrink" an object a little bit to avoid the situation that the object aligns with the grid; that is, an aligned object is treated as if it was contained by the aligned query window spanned by the object even when it equals the aligned query window.

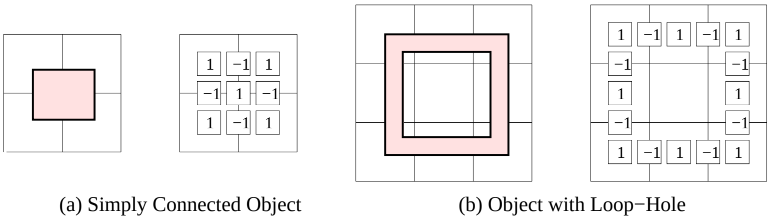


Fig. 3. Euler Histograms

Suppose that H is an Euler histogram for a set S of objects, and Q is an aligned query window.

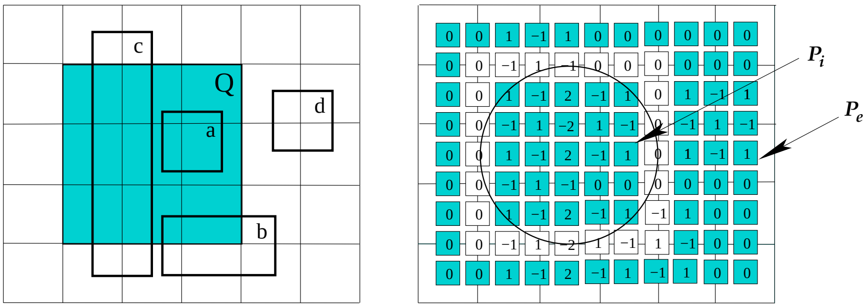
—$|S|$ denotes the total number of objects in S.

—N_{ds} denotes the number of objects in S which disjoint with Q.

—N_{nds} denotes the number of objects which non-disjoint with Q.

—P_i denotes the summation of all the bucket values inside Q (excluding the boundary of Q, shown in Figure 4).

Assume that S has only one *simply connected* object (i.e. without holes). The Euler formula [Harary 1969] implies that the summation of values from the buckets non-disjointing with the object is 1 (see Figure 3(a) for example). Therefore, $N_{nds} = P_i$; this together with $N_{nds} + N_{ds} = |S|$ yield the exact solutions for N_{nds} and N_{ds}.



(a) (b)

Fig. 4. Compute P_i and P_e

2.2 S-Euler Algorithm

In [Sun et al. 2002a], the non-disjoint relation is decomposed into 3 relations: overlap, contains, and contained, as depicted in Figure 1 since *equal* relation does not occur in an Euler histogram. Sun, Agrawal, and El Abbadi then proposed to use the histogram information outside a query window Q as well to summarize these 3 new relations. They extended the original Euler formula for processing an object with a *loop-hole* inside, which is caused by summarizing the outside of a query window but inside of an object. It can be shown that if we sum up the values of the buckets that an object intersects, then the result will be 0 if the object has a loop-hole inside (shown in Figure 3(b)). Here, a *loop-hole* means that there is at least one cell on the grid which is not contained by the object but encompassed by the "outside" boundary of the object; see Figure 3(b) for example. Note that any object studied in this paper is assumed without holes; however by investigating loop-holes we can effectively use the histogram information outside Q.

It has been shown that in an Euler histogram, the ov relation as illustrated in Figure 1(c) has to be separated into 1) *intersect* (the left figure in Figure 1(c)), and 2) *cross-over* (the right figure in Figure 1(c)), since they contribute differently to the outside of Q. Note that here, we abuse the original term "intersect" from [Egenhofer and Herring 1994; Sun et al. 2002a] for a term simplification; the "intersect" relation in [Egenhofer and Herring 1994; Sun et al. 2002a] corresponds to the "non-disjoint" relation in this paper. Although in this paper we aim to count the number of ov objects, we will have to first deal with the relations cross-over (cr) and intersect (it), and then add them together to obtain the number of ov objects.

—N_{cs} denotes the number of objects in S which Q contains;

—N_{it} denotes the number of objects in S which intersect Q.

—N_{cr} denotes the number of objects in S which cross over Q.

—N_{cd} denotes the number of objects in S by which Q is contained.

—P_e denotes the summation of all the bucket values outside Q (shown in Figure 4).

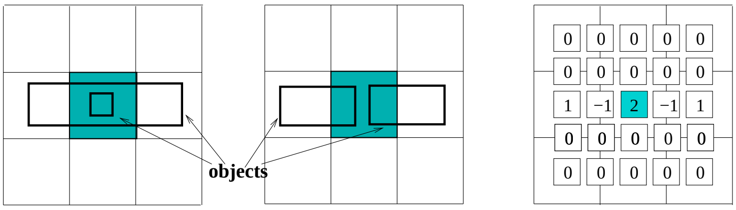
By a generalized Euler formula [Sun et al. 2002a], we have

$$P_e = N_{it} + 2N_{cr} + N_{ds} \tag{1}$$

Note that here, we have to count a cr object twice in (1). Clearly, $N_{nds} = N_{it} + N_{cr} + N_{cd} + N_{cs}$; this together with the equation (1) and the equations in the last subsection lead to:

$$\begin{aligned} N_{ds} &= |S| - P_i && (2)\\ N_{cr} &= \frac{1}{2}(P_i + P_e - |S| - N_{it}) && (3)\\ N_{cs} + N_{cd} &= \frac{1}{2}(P_i - P_e + |S| - N_{it}) && (4) \end{aligned}$$

Clearly, N_{ds} can be computed exactly since P_i can be computed from the histogram and $|S|$ is known. In the equations (3) and (4), there are 4 variables to be fixed. In fact, it is impossible to create more independent equations without introducing new variables. This is because the information in one Euler histogram is not enough to determine the 3 relations, cs, cd, and ov. For instance, in Figure 5 the two different scenarios (Figure 5(a) and Figure 5(b)) lead to the same histogram (Figure 5(c)). If we use the shadow area as a query window, we have no idea about what the scenario should be.



0	0	0	0	0
0	0	0	0	0
1	−1	2	−1	1
0	0	0	0	0
0	0	0	0	0

(a) contains=1, overlap=1 (b) contains=0, overlap=2 (c) Histogram for (a) and (b)

Fig. 5. A Counter Example

Motivated by this example, in S-Euler N_{cr} and N_{cd} are both removed from the equations (3) and (4) for approximation. Therefore, the two equations are just enough for the remaining two variables.

2.3 EulerApprox Algorithm

In this algorithm, N_{cr} is still assigned to 0 while N_{cd}, N_{it}, and N_{cs} remain in the two equations (3) and (4). Therefore, one more equation is needed; consequently, the following equation is added.

$$N_{it} + N_{cd} + N_{ds} = N_{cs}(B) + P_e(A) \tag{5}$$

As depicted in Figure 6, the whole space is split into two parts along one edge of Q. Here, $N_{cs}(B)$ is the number of objects contained in the shadow area B, which can be calculated exactly by the algorithm S-Euler. $P_e(A)$ is the summation of all bucket values in the interior of the shadow area A. It has been shown that (5) holds if we assume $N_{cr} = 0$ and the number of O_1 type objects equals the number of O_2 type objects.

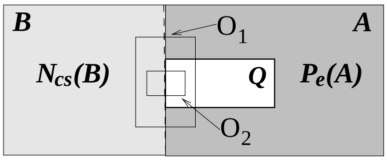


Fig. 6. EulerApprox

2.4 M-Euler Algorithm

This algorithm attempts to reduce the deficiency caused by those strong assumptions in S-Euler and EulerApprox by adopting multiple Euler histograms. In M-Euler, the object areas are divided into k ranges and construct one Euler histogram for the objects in each range.

Querying an Euler histogram H against an aligned query window Q in M-Euler proceeds as follows. If the area of each object involved in H is smaller (greater) than the area of Q then S-Euler algorithm is used (in the later $N_{cs} = 0$ instead of $N_{cd} = 0$). Otherwise, EulerApprox is applied.

2.5 Costs of Euler Histograms

An Euler histogram H with a resolution $n_1 \times n_2$ has $(2n_1 - 1) \times (2n_2 - 1)$ buckets, and each bucket stores an integer. Therefore, the storage space required by H is $O(n_1 \times n_2)$.

Since $|S|$ is given, S-Euler and EulerApprox run in constant time by solving the linear equations if P_i and P_e are already obtained; consequently, M-Euler takes $O(k)$ time where k is the number of histograms. In fact, P_i and P_e can be computed in constant time as follows if the *prefix-sum* techniques in [Ho et al. 1997] is applied to representing Euler histograms.

In H, $H(x, y)$ represents the value in the bucket (x, y) where (x, y) is a representation of a grid point, an edge, or a cell. For a $n_1 \times n_2$ grid, the grid points are $\{(i, j) : 1 \leq i \leq n_1 + 1, 1 \leq j \leq n_2 + 1\}$, an edge from the grid point (a, b) to the grid point (a', b') is represented by $(\frac{a+a'}{2}, \frac{b+b'}{2})$, and a cell, with four grid points (a, b), $(a + 1, b)$, $(a, b + 1)$, $(a + 1, b + 1)$, is represented by $(a + 0.5, b + 0.5)$. Note that for presentation simplification, we use non-grid points to represent a cell and an edge.

In the prefix-sum techniques, we use a cumulative representation $H^c(x, y)$ for each (x, y), that is,

$$H^c(x, y) = \sum_{x' \leq x, y' \leq y} H(x', y').$$

Note that we assume $H(x, y)$ equals zero if there is no entry in the histogram for (x, y). For a query window Q with the bottom-left corner (x_1, y_1) and the upper-right corner (x_2, y_2), the corresponding P_i and P_e are:

$$\begin{aligned} P_i \; = \; & H^c(x_2 - \frac{1}{2}, y_2 - \frac{1}{2}) + H^c(x_1, y_1) - \\ & H^c(x_1, y_2 - \frac{1}{2}) - H^c(x_2 - \frac{1}{2}, y_1), \end{aligned} \tag{6}$$

$$P_e = |S| - H^c(x_2, y_2) + H^c(x_1 - \frac{1}{2}, y_2) + H^c(x_2, y_1 - \frac{1}{2}) - H^c(x_1 - \frac{1}{2}, y_1 - \frac{1}{2}). \tag{7}$$

2.6 Motivation and Problem Statement

The strong assumptions that $N_{cr} = 0$ or/and $N_{cd} = 0$ in S-Euler and EulerApprox greatly downgrade the performance of the two algorithms if the underlying data do not follow the assumptions.

M-Euler aims to remove the disadvantages of S-Euler and EulerApprox by grouping objects together according to their areas. As rectangles with different shapes may have the same area, the disadvantages of S-Euler and EulerApprox cannot be removed effectively by M-Euler.

The full paper version of [Sun et al. 2002a] proved that in the *worst case*, any cell density based histogram requires $\Omega(n_1^2 \times n_2^2)$ storage space to count N_{cs} exactly for aligned windows with respect to a $n_1 \times n_2$ resolution. Note that $\Omega(n_1^2 \times n_2^2)$ is quadratic with respect to $n_1 \times n_2$ and the number of buckets is $(2n_1 - 1) \times (2n_2 - 1)$. Therefore, it is impossible to develop a cell density based spatial histogram technique to be always practical (i.e., storage space linearly proportional to the number of cells) that guarantees exact solutions to N_{cs}.

Problem Statement. Motivated by these, in this paper we will present a multiscale Euler histogram technique with the guarantee of exact solutions to the aligned windows, which may be practical for many real applications where the number of different scales is small. Then, we will also present another multiscale Euler histogram with high accuracy of approximation (though no guarantee of exact solutions), which is always practical. Our new techniques remove the assumptions that $N_{cr} = 0$ or/and $N_{cd} = 0$.

Note that in our investigation we assume that a resolution is pre-given and the aligned windows are defined on a pre-given resolution. Such a pre-given resolution may be from the semantics of applications (e.g., the resolution of the longitude and latitude of a digital map) or from a resolution requirement of applications (i.e., specified according to users' requirements).

3. MULTISCALE HISTOGRAMS

In this section, we will present a multiscale paradigm to construct Euler histograms which can guarantee the exact solutions for N_{cs}, N_{cd}, N_{cr}, N_{it}, and N_{ds} for aligned windows. Note that in this section and the next section, we will focus only on aligned query windows with respect to a given grid (resolution); thus, the expression is abbreviated to "a query window" in these two sections whenever no ambiguities. We begin with the framework.

3.1 Construction Techniques

The basic idea of our multiscale paradigm is to group the objects together according to their *scales*. An object (rectangle) has the *scale* (w, h) with respect to a grid (resolution) if its horizontal edge crosses w cells and its vertical edge crosses h cells (see Figure 2(c) for example). The *scale* of a query window refers to the scale of its corresponding rectangle.

The following theorem characterises a relationship between the scales and the topological relations for a given query window. The theorem can be immediately verified.

THEOREM 3.1. *Suppose that Q is a query window with the scale (i, j), and D is an object with the scale (w, h) (both scales are referred to the same grid).*

—*If Q contains D, then $w \leq i$ and $h \leq j$.*

—*If D crosses over Q, then ($w \leq i$ and $h \geq j + 2$) or ($w \geq i + 2$ and $h \leq j$).*

—*If Q is contained by D, then $w \geq i + 2$, and $h \geq j + 2$.*

The theorem below is the key to the correctness of our algorithm. It states that a histogram based on the objects with 4 "adjacent" scales can guarantee the exact solutions.

THEOREM 3.2. *Suppose that H is an Euler histogram with a $n_1 \times n_2$ resolution, such that the objects involved in H have at most 4 scales, (w, h), $(w+1, h)$, $(w, h+1)$, and $(w + 1, h + 1)$. Then, H can provide the exact solutions to N_{ds}, N_{it}, N_{cs}, N_{cd} and N_{cr} for a query window Q.*

PROOF. Suppose that the scale of Q is (i, j) ($1 \leq i \leq n_1$ and $1 \leq j \leq n_2$). Clearly, N_{ds} can be obtained exactly from equation (2). Below are the three cases by comparing (w, h) with (i, j) while querying H against Q:

Case 1:. $w \leq i$ and $h \leq j$ (depicted in Figure 7(a)).

Case 2:. ($w > i$ and $h \leq j$) or ($w \leq i$ and $h > j$) (depicted in Figure 7(b)).

Case 3:. $w > i$ and $h > j$ (depicted in Figure 7(c)).

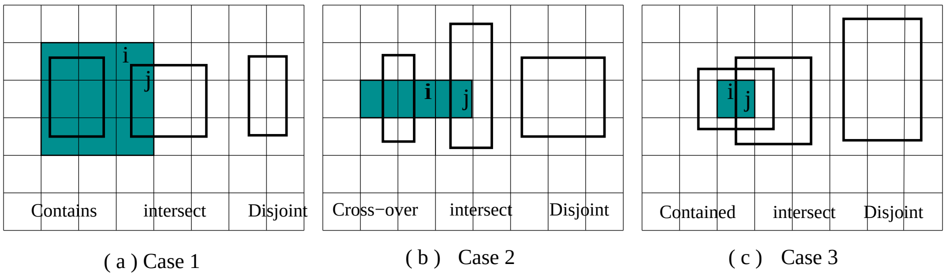


Fig. 7. Three Cases by Comparing Q with (w, h)

According to Theorem 3.1, in case 1 no object in this histogram can cross over Q, nor Q is contained by an object. That is, $N_{cr} \equiv 0$ and $N_{cd} \equiv 0$. Clearly, the remaining two variables N_{cs} and N_{it} can be fixed from the equations (3) and (4).

In case 2, clearly there is no cd relation nor cs relation; that is, $N_{cd} \equiv 0$ and $N_{cs} \equiv 0$. Again, the two remaining variables can be fixed by the two equations.

In case 3, based on Theorem 3.1 there is no cr relation nor cs relation; that is, $N_{cr} \equiv 0$ and $N_{cs} \equiv 0$. Thus, the two remaining variables can also be fixed by the two equations. □

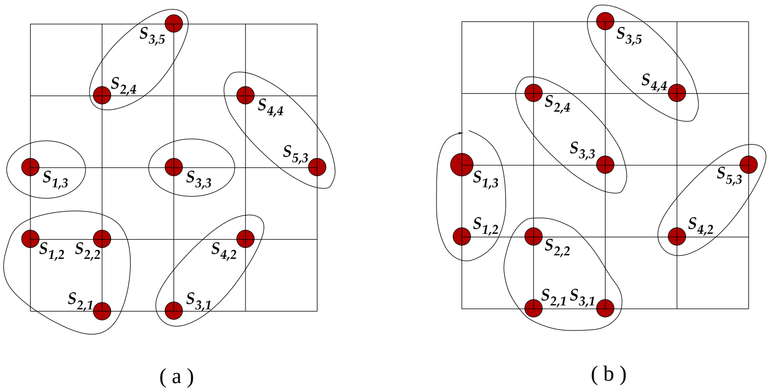


Fig. 8. Different Sets of Subsets

Let S denote a set of objects. In the rest of the paper, $Q_{i,j}$ always denotes a query window with the scale (i,j); $S_{w,h}$ denotes the set of objects, in S, with the scale (w,h). Let $\widehat{S} = \{S_{w,h} : 1 \leq w \leq n_1, 1 \leq h \leq n_2, |S_{w,h}| \neq 0\}$.

To describe our techniques, each dataset $S_{w,h} \in \widehat{S}$ is mapped into a grid point (w,h) on the $n_1 \times n_2$ grid. $B_{\widehat{S}}$ denotes the set of subsets of $\widehat{S}$ with the property that each of such subsets of $\widehat{S}$ consists of the datasets whose corresponding grid points lie within one cell. For example, regarding Figure 8, $B_{\widehat{S}}$ is

$$\{\{S_{1,2}, S_{2,1}, S_{2,2}\}, \{S_{1,2}, S_{2,1}\}, \{S_{1,2}, S_{2,2}\}, \{S_{2,1}, S_{2,2}\},$$
$$\{S_{1,2}\}, \{S_{2,1}\}, \{S_{2,2}\}, \{S_{2,1}, S_{2,2}, S_{3,1}\}, ...\}.$$

It is immediate that $|B_{\widehat{S}}| = O(|\widehat{S}|)$. Note that $B_{\widehat{S}}$ is closed under intersection if we add $\emptyset$ to $B_{\widehat{S}}$. For a given S and a given resolution $n_1 \times n_2$ of S, $\widehat{S}$ is unique, and $B_{\widehat{S}}$ is also unique.

An object D is *involved* in an element ξ of $B_{\widehat{S}}$ if D is in a dataset in ξ; for instance, an object is involved in $\{S_{2,1}, S_{2,2}, S_{1,2}\}$ if the object is in one of $S_{2,1}$, $S_{2,2}$, and $S_{1,2}$. A subset of $B_{\widehat{S}}$ is *well-separated* if every pair of elements in the subset do not share a common dataset; for instance, the subset, $\{\{S_{1,2}, S_{2,2}\}, \{S_{2,1}, S_{3,1}\}\}$, of $B_{\widehat{S}}$ is well-separated. However, the subset, $\{\{S_{1,2}, S_{2,2}\}, \{S_{2,2}, S_{2,1}\}\}$, of $B_{\widehat{S}}$ is not well-separated since the two subsets, $\{S_{1,2}, S_{2,2}\}$ and $\{S_{2,2}, S_{2,1}\}$, of $\widehat{S}$ share the common dataset $S_{2,2}$.

Theorem 3.2 states that a set of objects, which are involved in one element in $B_{\widehat{S}}$, can be represented by the Euler histogram to support the exact solutions. Our algorithm is to find a well-separated subset Λ of $B_{\widehat{S}}$ such that $\widehat{S}$ is covered by Λ; it consists of the following 3 steps.

Figure 8(a) illustrates such a Λ with 6 elements, where each element in Λ is "circled". Querying the set of histograms constructed by MESA for a query window may be easily done by querying each histogram with respect to the three cases as described in the proof of Theorem 3.2, respectively; then adding up the values over all the histograms gives the global N_{cr}, N_{it}, N_{cs}, N_{cd}, and N_{ds}. According to Theorem 3.2, the algorithm MESA is correct; that is, the histograms constructed

Algorithm 1 Multi-Scale Exact Algorithm (MESA)

```
INPUT. a data set $S$ and a given resolution (the $n_1 \times n_2$ grid).
OUTPUT. a set of histograms with respect to the given resolution.
Step 1. Scan $S$ to generate $\widehat{S}$ and $B_{\widehat{S}}$ according to the given resolution
Step 2. Find a well-separated subset $\Lambda$ of $B_{\widehat{S}}$ such that $\widehat{S}$ is covered by $\Lambda$.
Step 3. For each element $\xi \in \Lambda$, construct the Euler histogram $H_\xi$, with respect to
    the resolution, to represent the objects involved in $\xi$.
```

can provide the exact solutions to these five values for (aligned) window queries. Since querying each histogram takes constant time (see section 2), querying $|\Lambda|$ Euler histograms constructed by MESA takes $O(|\Lambda|)$ time.

Note that the Step 1 is immediate. An implementation of Step 3 has been briefly described in section 2; the details may be found in [Beigel and Tanin 1998].

Suppose that Λ is chosen in the Step 2. The number ($|\Lambda|$) of histograms produced by MESA is called the *thickness* of the set of histograms. Clearly, there may be many well-separated subsets of $B_{\widehat{S}}$ to be chosen as an output of the Step 2. Figure 8 shows two different well-separated subsets of $B_{\widehat{S}}$; both cover $\widehat{S}$. One (Figure 8(a)) gives the thickness 6 and another (Figure 8(b)) gives the thickness 5. In this example, 5 is the minimum thickness.

Note that a multiscale histogram constructed by MESA requires $O(|\Lambda| \times n_1 \times n_2)$ space for the resolution $n_1 \times n_2$. The minimization of such a $|\Lambda|$ means the minimization of the histogram storage space for a given resolution, as well as the query processing costs.

3.2 Minimization of Thickness

According to the construction of $B_{\widehat{S}}$, it is immediate that for any such Λ produced by MESA, $\frac{k}{4} \leq |\Lambda| \leq k$ where $k = |\widehat{S}|$. The minimization problem is formally defined below.

Optimal Data Partitioning Problem (ODP)

Instance: Suppose that $\widehat{S}$ and $B_{\widehat{S}}$ are given as above.

Question: find a well-separated subset Λ of $B_{\widehat{S}}$, such that $\widehat{S}$ is covered by Λ and $|\Lambda|$ is minimized.

Recall that each element in $B_{\widehat{S}}$ corresponds to the grid points on one cell. ODP is a special case of the 4-set cover problem [Duh and Fürer 1997]; the 4-set cover problem is well-known NP-hard in general. Although a very special case of the 4-set cover problem, unfortunately ODP is still NP-hard. The proof of Theorem is not trivial and will be shown in the Appendix.

THEOREM 3.3. *ODP is NP-hard.*

An Approximate Algorithm to ODP. Below in Algorithm 2, we present an approximate algorithm to solve ODP.

Note that in Step 2, after removing the elements from $B_{\widehat{S}}$ with an intersection to an element selected in Step 1, each remaining element of $B_{\widehat{S}}$ has the cardinalities at most 2. The remaining $B_{\widehat{S}}$ can be viewed as a graph G, where a vertex corresponds

Algorithm 2 Minimizing the Thickness (MT)

INPUT. $B_{\widehat{S}}$ and $\widehat{S}$.

OUTPUT. an approximate solution λ to ODP.

Step 1. Iteratively, generate a well-separated subset λ_1 of $B_{\widehat{S}}$ such that the cardinality of each selected element from $B_{\widehat{S}}$ in each iteration is the maximum among the qualified elements but not smaller than 3.

Step 2. Remove from $B_{\widehat{S}}$ the elements intersecting an element chosen in Step 1. Run the graph maximum matching algorithm to obtain a subset λ_2 of the remaining $B_{\widehat{S}}$.

Step 3. Return $\lambda_1 \cup \lambda_2$ together with the uncovered singletons in $B_{\widehat{S}}$.

to a remaining singleton, and each edge corresponds to a remaining element with the cardinality 2. Thus, we can run the maximum matching algorithm to get a maximum matching. Each edge in the maximum matching corresponds to an element in the remaining $B_{\widehat{S}}$, which will be chosen in Step 2. It is immediate that Step 2 takes the dominant costs and runs in $O(|\widehat{S}^{1.5}|)$ [Micali and Vazirani 1980].

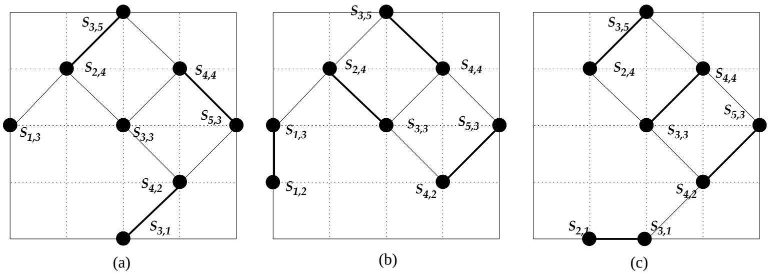


Fig. 9. Example Graphs in Step 2

With respect to the example in Figure 8, Step 1 can select only one element. Suppose that we choose $\{S_{1,2}, S_{2,1}, S_{2,2}\}$ in Step 1, then after removing the relevant elements from $B_{\widehat{S}}$, we obtained a graph G with 8 vertices and 9 edges as depicted in Figure 9(a). For this graph, a maximum matching can have only 3 edges. Suppose that $\{(S_{2,4}, S_{3,5}), (S_{4,4}, S_{5,3}), (S_{4,2}, S_{3,1})\}$ is output as a maximum matching in Step2 as depicted in Figure 9(a) by thick edges. Then, $S_{1,3}$ and $S_{3,3}$ are chosen in Step 3. In this case, Λ is what is depicted in Figure 8(a).

The *semi-local* optimization technique in [Duh and Fürer 1997] may be used to refine the result produced by the algorithm MT; this can guarantee [Duh and Fürer 1997] the approximate ratio $\frac{19}{12}$. The basic idea of this technique is to iteratively improve the quality of current approximation result by a semi-local change in combining with the algorithm MT. The interested readers may refer to [Duh and Fürer 1997] for the details. Below we show one example.

As depicted in Figure 8(a), suppose that the circled elements are the output of the algorithm MT. Running the semi-local optimization algorithm, we need to choose a replacement to $\{S_{1,2}, S_{2,1}, S_{2,2}\}$. In this example, $\{S_{2,1}, S_{2,2}, S_{3,1}\}$ and

$\{S_{1,3}, S_{1,2}, S_{2,2}\}$ are the options; both of them can guarantee the minimum thickness 5 by the semi-local optimization algorithm. Figure 8(b) shows such an result where the first option replacement is adopted and the remaining graph is depicted in Figure 9(b). Note that Figure 9(c) shows the remaining graph if the second replacement option is used; that is, $\{S_{1,3}, S_{1,2}, S_{2,2}\}$ is chosen.

Note that the Semi-local optimization may have $O(|L|^3)$ iterations in the worst case where L is the number of elements in $B_{\widehat{S}}$ with cardinality 3; each iteration runs in $O(|\widehat{S}|^{1.5})$ time. However, in the real datasets used in our experiments, we found that even the algorithm MT can generate the optimal thickness; this means no iteration. For instance, for the Texas road segments data of US Census Tiger [TIGER 2000] with the 360 × 180 resolution, the minimum thickness 13 can be computed by our algorithm MT, while there are 35 different scales. Further, for the datasets in our experiments semi-local optimization converges very fast. The performance of MESA will be evaluated in section 9.

4. MULTISCALE HISTOGRAMS WITH A FIXED SPACE

The exact algorithm proposed in the last section suits for datasets with small number ($|\widehat{S}|$) of scales for a given resolution. When $|\widehat{S}|$ is large and the storage space is limited, MESA is not always applicable. Besides, we observed that in many real world datasets the majority of objects have similar scales at a reasonable resolution while the total number of outliers (objects) may be very small; thus, it is not economic to use more than one histogram to approximate a small set of objects. Motivated by these, in this section we will present an effective paradigm to construct a set of histograms, such that the number of histograms to be used is $k + 1$ for a fixed k.

The main idea of our algorithm is to construct k histograms which can provide the exact solutions for the objects involved, while the remaining objects are all put into the last histogram which cannot guarantee the exact solutions. Intuitively, less objects are involved in the last histogram, higher accuracy of approximation may be globally expected on average. Therefore, in our algorithm we aim to allocate the objects to the first k histograms as many as possible while retaining the property of providing exact solutions. Below is a description of our algorithm. For a set Λ of subsets of $\widehat{S}$, $||\Lambda||$ denotes the number of objects involved in Λ.

Algorithm 3 Multi-scale Approximate Algorithm (MAPA)

INPUT. a set S of objects in 2-d space, a given resolution ($n_1 \times n_2$ grid), and an integer $k + 1$.

OUTPUT. $k + 1$ histograms with respect to the resolution.

Step 1. Scan S to generate $\widehat{S}$ and $B_{\widehat{S}}$ according to a given resolution (the $n_1 \times n_2$ grid).

Step 2. Find a well-separated subset Λ of $B_{\widehat{S}}$ such that $|\Lambda| = k$ and $||\Lambda||$ is maximized.

Step 3. For each element $\xi \in \Lambda$, construct the Euler histogram H_ξ, with the $n_1 \times n_2$ resolution, to represent the objects involved in ξ.

Step 4. Construct the Euler histogram H_{last} for the objects not involved in Λ.

In MAPA, the Steps 1, 3, 4 are the same as those in the algorithm MESA. In the subsection 4.1, we will present our results for Step 2. As with the algorithm MESA, the first k histograms generated by the algorithm MAPA can guarantee the exact solutions for N_{cr}, N_{it}, N_{cs}, N_{cd}, and N_{ds} restricted to the objects involved in Λ. In subsection 4.2, we will present a new algorithm to summarize the objects involved in the last histogram H_{last}.

4.1 Data Partition

The optimization problem in Step 2 may be formally described below.

Weighted k-Partitioning Problem (WkP)
Instance: Suppose that $\widehat{S}$ and $B_{\widehat{S}}$ are given as described in section 3, and an integer k is given.
Question: find a well-separated subset Λ of $B_{\widehat{S}}$ such that $|\Lambda| = k$ and $||\Lambda||$ is maximized.

Theorem 4.1. *WkP is NP-hard.*

Proof. A special case of WkP, where each dataset in $\widehat{S}$ has the same number of objects, is more general than the corresponding decision problem of ODP. Therefore, WkP is NP-hard. □

A similar problem to WkP, called "Max k-cover" problem, was investigated in [Feige 1998]. In Max k-cover problem, a collection F of subsets of $V = \{v_1, ..., v_n\}$ is given; select k elements (subsets) from F such that their union has maximum cardinality. Note that Max k-cover problem is different to WkP where each v_i has a weight while each subset of V has the cardinality up-to 4. It has been shown [Feige 1998] that the Max k-cover problem may be solved by an approximate algorithm with the approximate ratio $1 - \frac{1}{e}$. Below, we show that a greedy heuristic to approach WkP has the same performance.

Algorithm 4 GreedyWkP($B_{\widehat{S}}, k, \Lambda$)

```
Sort the elements in $B_{\widehat{S}}$ decreasingly based on the number of objects involved in each element;
$\Lambda \leftarrow \emptyset$;
while $|\Lambda| \neq k$ and $|B_{\widehat{S}}| \neq 0$ do
    get the 1st element $\xi$ from $B_{\widehat{S}}$;
    remove the element from $B_{\widehat{S}}$ intersecting $\xi$;
    $\Lambda \leftarrow \Lambda \cup \{\xi\}$;
end while
```

Theorem 4.2. *The algorithm GreedyWkP guarantees the approximate ratio not less than* $1 - \frac{1}{e}$.

Proof. Let Λ_{opt} denote the solution to WkP, that is $|\Lambda_{opt}| \leq k$ and $||\Lambda_{opt}||$ is maximized. Let ξ_i be the element chosen in the ith iteration of the algorithm GreedyWkP and m_i denote the total number of data objects involved in ξ_i. Then, for $1 \leq i \leq k$, $m_i \geq \frac{||\Lambda_{opt}|| - \sum_{j=1}^{i-1} m_j}{k}$. This immediately implies that

$$\sum_{j=1}^{k} m_j \geq ||\Lambda_{opt}||(1 - (1 - \frac{1}{k})^k) \tag{8}$$

□

According to the definition of $B_{\widehat{S}}$, there are a constant number of subsets (of $\widehat{S}$) in $B_{\widehat{S}}$ intersecting another subset in $B_{\widehat{S}}$. Thus, the dominant cost of GreedyWkP is in sorting $B_{\widehat{S}}$. Recalling that $|B_{\widehat{S}}| = O(|\widehat{S}|)$, GreedyWkP runs in time $O(n \log n)$ where $n = |\widehat{S}|$.

4.2 Summarizing the Last Histogram

In this subsection, we present a new technique to summarize the object set involved in the last histogram H_{last}. It effectively uses scales in combining with the Euler histogram.

Given an object scale (w, h) and a query window $Q_{i,j}$ with respect to the $n_1 \times n_2$ resolution, the 3 cases in Theorem 3.1 can be further divided into the following 5 cases:

Case 1. $w \leq i$ and $h \leq j$ - at most 3 relations: cs, it, and ds.

Case 2. $w = i + 1$ or $h = j + 1$ - at most 2 relations: it and ds.

Case 3a. $w \geq i + 2$ and $h \leq j$ - at most 3 relations: cr, it, and ds.

Case 3b. $w \leq i$ and $h \geq j + 2$ - at most 3 relations: cr, it, and ds.

Case 4. $w \geq i + 2$ and $h \geq j + 2$ - at most 3 relations: cd, it, and ds.

We should be able to estimate the occurring probabilities against the 5 relations (cr, it, cs, cd, and ds), respectively, in the object scale (w, h) with respect to $Q_{i,j}$. For a given case and a given topological relation, we will calculate the ratio of the number of possible grid points to be used as the *bottom-left* corner of an object with the scale (w, h) to form the relation over the number of possible grid points to be used as the bottom-left corner of an object with the scale (w, h).

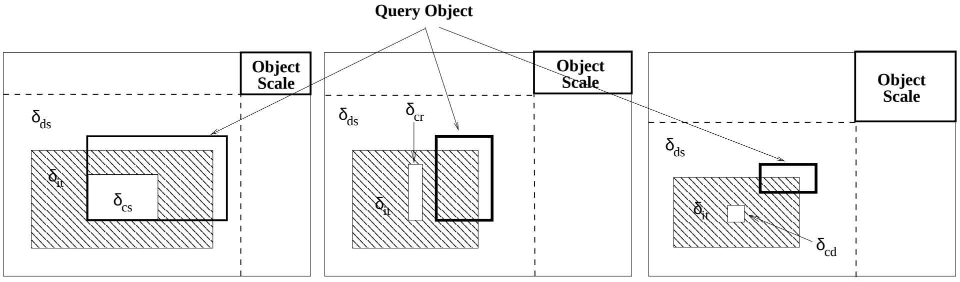
As depicted by the rectangular areas in Figure 10, we use δ_{cr}, δ_{it}, δ_{cs}, δ_{cd}, and δ_{ds} to denote the number of grid points, possibly used as bottom-left object corners for the 5 relations, respectively. Note that in Figure 10, we illustrate only three cases: Case 1, Case 3a, and Case 4. Case 3b is similar to Case 3a; and Case 2 is similar to all these three cases but without the white area in the middle.

Below we present the detailed formulae to identify those rectangles in Figure 10 with respect to each case. The formulae may be immediately obtained by elementary geometry; thus, we omit the deduction details from this paper. Note that the size and position of each rectangle area in Figure 10 not only depend on the scales (w, h) and (i, j) but also depend on the position of $Q_{i,j}$.

Suppose that a query rectangle $Q_{i,j}$ with the bottom-left corner (q_x, q_y), and a rectangle is represented by $\{(x, y), (a, b)\}$ where (x, y) is the bottom-left corner and (a, b) represents (width, height).

Case 1:

In this case, $\delta_{cr} \equiv 0$ and $\delta_{cd} \equiv 0$, while the others can be calculated as follows.



(a) Case 1 (b) Case 3a (c) Case 4

Fig. 10. Different cases

δ_{cs} is the number of grid points in the rectangle $r_{cs} = \{(x_{cs}, y_{cs}), (a_{cs}, b_{cs})\}$ (the white area in Figure 10(a)). Here,

—$x_{cs} = q_x$ and $y_{cs} = q_y$;
—$a_{cs} = i - w$, and $b_{cs} = j - h$.

Note that "in the rectangle" also include the points on the edge; this is also applicable to the cases below. Note $a_{cs} = 0$ and $b_{cs} = 0$ imply there is only one point (x_{cs}, y_{cs}).

δ_{it} is the number of grid points in the rectangle (the one with slash lines in Figure 10(a)) $r_{it} = \{(x_{it}, y_{it}), (a_{it}, b_{it})\}$ but exclude the points in r_{cs}. Note that in this case, r_{it} is getting smaller when an edge of $Q_{i,j}$ approaches a boundary of the grid. Below is a precise formula.

—$x_{it} = 1 + \max\{0, q_x - w\}$ and $y_{it} = 1 + \max\{0, q_y - h\}$;
—$a_{it} = q_x + i - 1 - x_{it} - \max\{0, q_x + i + w - n_1 - 2\}$ and $b_{it} = q_y + j - 1 - y_{it} - \max\{0, q_y + j + h - n_2 - 2\}$.

Note that an object with the scale $(1, 1)$ never intersects $Q_{i,j}$; consequently $\delta_{it} \equiv 0$ in this case. This can be reflected by the formula. In fact we can verify that r_{it} is always the same as r_{cs} when $w = 1$ and $h = 1$; and thus δ_{it} is calculated as 0.

δ_{ds} is the number of grid points in the rectangle $r_{ds} = \{(x_{ds}, y_{ds}), (a_{ds}, b_{ds})\}$ (the one bounded by dashed line in Figure 10) except the grid points in r_{it}. Here,

—$x_{ds} = 1$ and $y_{ds} = 1$;
—$a_{ds} = n_1 - w$ and $b_{ds} = n_2 - h$.

Note that this rectangle is determined only by the object scale (w, h). In case 1, the probabilities for cs, it, and ds, respectively, are

$$\begin{aligned} \rho^1_{cs} &= \frac{\delta_{cs}}{\delta_{cs} + \delta_{it} + \delta_{ds}}, \ \rho^1_{it} = \frac{\delta_{it}}{\delta_{cs} + \delta_{it} + \delta_{ds}}, \\ \rho^1_{ds} &= \frac{\delta_{ds}}{\delta_{cs} + \delta_{it} + \delta_{ds}}. \end{aligned}$$

It can be immediately verified that the rectangle r_{it} always includes the rectangle r_{cs}, and is always included by r_{ds}. Therefore, the computation of δ_{cs}, δ_{it}, and δ_{ds} is simple. We first calculate δ_{cs} from r_{cs}. We then calculate δ_{it} as the total points in r_{it} minus δ_{cs}, and calculate δ_{ds} as the total points in r_{ds} minus δ_{cs} and minus δ_{it}. Therefore, ρ^1_{cs}, ρ^1_{it}, and ρ^1_{ds} can be calculated in constant time.

Case 2:

In this case, $\delta_{cs} \equiv 0$, $\delta_{cr} \equiv 0$, and $\delta_{cd} \equiv 0$, while δ_{it} and δ_{ds} can be calculated by the above formula. By similar reasons, $\rho^2_{it} = \frac{\delta_{it}}{\delta_{it}+\delta_{ds}}$ and $\rho^2_{ds} = \frac{\delta_{ds}}{\delta_{it}+\delta_{ds}}$ can be computed in constant time. Note that we can more precisely handle this case by dividing it into two sub-cases: a) $a = i+1$ and $b \neq j+1$, and b) $b = j+1$. However, our experiments show that we gain very marginally by doing this. Therefore, we omit these from the paper.

Case 3a:

In this case, $\delta_{cs} \equiv 0$ and $\delta_{cd} \equiv 0$. We can calculate δ_{cr} as follows.

—$\delta_{cr} = 0$ if $q_x = 1$ or $q_x + i = n_1 + 1$, otherwise

—δ_{cr} is the number of grid points in the rectangle $r_{cr} = \{(x_{cr}, y_{cr}), (a_{cr}, b_{cr})\}$ (white rectangle in Figure 10(b)). Here,

  —$x_{cr} = 1 + \max\{0, q_x + i - w\}$ and $y_{cr} = q_y$;

  —$a_{cr} = q_x - 1 - x_{cr} - \max\{0, q_x + w - n_1 - 2\}$ and $b_{cr} = j - h$.

Note that the rectangles for δ_{it} and δ_{ds} can be calculated by the same formulae as those in Case 1. Again, it can be immediately verified that the rectangle r_{it} always includes r_{cr} and is included by r_{ds}. Similarly, $\rho^{3a}_{cr} = \frac{\delta_{cr}}{\delta_{cr}+\delta_{it}+\delta_{ds}}$, $\rho^{3a}_{it} = \frac{\delta_{it}}{\delta_{cr}+\delta_{it}+\delta_{ds}}$, and $\rho^{3a}_{ds} = \frac{\delta_{ds}}{\delta_{cr}+\delta_{it}+\delta_{ds}}$ can be computed in constant time, respectively.

Case 3b:

Every thing can be viewed as a reflectional image, by the diagonal of the query rectangle, of case 3a; and thus may be calculated in a similar way to those in Case 3a. Therefore, ρ^{3b}_{cr}, ρ^{3b}_{it}, and ρ^{3b}_{ds} can be computed in constant time.

Case 4:

In this case, $\delta_{cs} \equiv 0$ and $\delta_{cr} \equiv 0$. Clearly, $\delta_{cd} = 0$ if $q_x = 1$ or $q_y = 1$ or $q_x+i = n_1+1$ or $q_y+j = n_2+1$; otherwise the rectangle $r_{cd} = \{(x_{cd}, y_{cd}), (a_{cd}, b_{cd})\}$ for calculating δ_{cd} is as follows.

—$x_{cd} = 1 + \max\{0, q_x + i - w\}$ and $y_{cd} = 1 + \max\{0, q_y + j - h\}$;

—$a_{cd} = q_x - 1 - x_{cd} - \max\{0, q_x + w - n_1 - 2\}$ and $b_{cd} = q_y - 1 - y_{cd} - \max\{0, q_y + h - n_2 - 2\}$.

Note that the rectangles for δ_{it} and δ_{ds} can be calculated by the same formulae as those in Case 1. It can be immediately verified that the rectangle r_{it} includes r_{cd} but is included by r_{ds}. Similarly, ρ^3_{ds}, ρ^3_{it}, and ρ^3_{cd} may be computed in constant time.

Note that for a given query window $Q_{i,j}$ and a set of m objects with the scale (w, h), we can estimate N_{cr}, N_{it}, N_{cs}, N_{cd} purely by the above probabilities; that is, $N'_{cr} = \rho_{cr}m$, $N'_{it} = \rho_{it}m$, $N'_{cs} = \rho_{cs}m$, $N'_{cd} = \rho_{cd}m$, and $N'_{ds} = \rho_{ds}m$. Since we assume that $m_{w,h} = |S_{w,h}|$ is recorded for each $S_{w,h}$, we can immediately calculate the above estimation. Therefore, by summing up all the above estimates, respectively, we can get the global estimation of N_{cr}, N_{it}, N_{cs}, N_{cd} and N_{ds}.

The pure probability approach above has two limitations: 1) the running time is $O(k)$ (k is the number of object scales) which is not necessary a constant, 2) it does not make the use of the advantages of an Euler histogram.

Now we present the algorithm Prob with constant time, combining the probability approach above with the Euler histogram. The basic idea is to divide the objects involved in H_{last} into the possible 5 cases (groups) as stated above; then we use an average rectangle to approximately represent the objects in each case together with the number of objects. Thus, for each case we can use constant time to compute the conditional possibility. Below (Algorithm 5) is a description of Algorithm Prob.

Algorithm 5 Prob (for a given Q)

```
$\alpha := 0$; $\beta := 0$; $\mu := 0$; $\gamma := 0$;
for each Case $i$ ($i \in \{1, 2, 3a, 3b, 4\}$ do
    calculate $m_i$; //the number of objects in this case;
    calculate $\bar{w}_i$ and $\bar{h}_i$; // average width and height;
    calculate $\rho^i_{cr}$, $\rho^i_{it}$, $\rho^i_{cs}$, and $\rho^i_{cd}$ against $\bar{w}_i$ and $\bar{h}_i$;
    $\alpha := \alpha + m_i\rho^i_{cr}$; $\beta := \beta + m_i\rho^i_{it}$; $\mu := \mu + m_i\rho^i_{cs}$; $\gamma := \gamma + m_i\rho^i_{cd}$;
end for
if $\gamma + \mu = 0$ then
    $N_{cs} := 0$; $N_{cd} := 0$;
    calculate $N_{cr}$ and $N_{it}$ from (3) and (4);
else
    get $N_{cr}$ and $N_{it}$ from (3) by $N_{cr} : N_{it} = \alpha : \beta$;
    get $N_{cd}$ and $N_{cs}$ from (4) by $N_{cs} : N_{cd} = \mu : \gamma$;
end if
```

Note that in Prob, if each m_i, $\bar{w}_i$ and $\bar{h}_i$ can be calculated in constant time then the whole algorithm will run in constant time. Below we show a pre-fix data structure to accommodate such a request.

Time Complexity of Algorithm Prob

We apply the prefix-sum technique to representing $\{m_{a,b} : 1 \leq a \leq n_1, 1 \leq b \leq n_2\}$; for $1 \leq a \leq n_1$ and $1 \leq b \leq n_2$, $m'_{a,b} = \sum_{1 \leq w \leq a, 1 \leq h \leq b} m_{w,h}$.

Let $w_{a,b}$ and $h_{a,b}$ denote the total widths and total heights of the objects in the scales $[1, a] \times [1, b]$, respectively. Besides H_{last}, in algorithm Prob we also pre-store

$$\{(m'_{a,b}, w_{a,b}, h_{a,b}) : 1 \leq a \leq n_1, 1 \leq b \leq n_2\}.$$

By similar methods to those in section 2.5, the total width, total height, and total number can be computed, respectively, for each case in constant time; then we can calculate the average widths and heights accordingly. Consequently, the algorithm Prob runs in constant time.

Note that the storage space to execute algorithm Prob is $(2n_1-1)(2n_2-1)+3n_1n_2$ which is about 75% more than the storage space ($(2n_1 - 1)(2n_2 - 1)$) for one Euler histogram.

Since querying every histogram runs in constant time, querying a set of histograms generated in MAPA runs in time $O(k)$ for a window query, where k is the number of histograms.

4.3 Remarks

Intuitively, the larger k is, the more accurate summarization results are for a given resolution. Our performance study conforms this. Consequently, given a storage space bound C and a resolution $n_1 \times n_2$, the number $k+1$ of histograms can be calculated from the following formula.

$$((k+1)(2n_1 - 1)(2n_2 - 1) + 3n_1 \times n_2)L = C. \tag{9}$$

Here, L is the space required for storing an integer. Note that the scales used in Algorithm 5 do not have to be the same as the scales for the resolution $n_1 \times n_2$. If this is the case, the formula (9) for calculating k needs to be adjusted accordingly.

5. NON-ALIGNED WINDOW QUERIES

In many applications, window queries from users are not necessarily aligned with a given grid. We propose to use the following two approximation techniques to resolve this problem. The first technique is to find the most similar window to a given query window, while the second technique is based an interpolation between the two closest windows.

5.1 Similar Query Window

Given two query windows Q and Q', let Q be represented by $\{(q_x, q_y), (u_x, u_y)\}$ and Q' be represented by $\{(q'_x, q'_y), (u'_x, u'_y)\}$. Here, (q_x, q_y) is the left-bottom corner of Q, and (u_x, u_y) is the upper-right corner of Q, while $\{(q'_x, q'_y), (u'_x, u'_y)\}$ are the corresponding parameters for Q'. We use the total distances, below, between the left-bottom corners and upper-right corners, respectively, to measure the similarity of the two rectangles.

$$SM(Q, Q') = \sqrt{(q_x - q'_x)^2 + (q_y - q'_y)^2} + \sqrt{(u_x - u'_x)^2 + (u_y - u'_y)^2} \tag{10}$$

Clearly, Q is identical to Q' iff $SM(Q, Q') = 0$. For a given query window Q, we propose to find an aligned window Q' such that $SM(Q, Q')$ is minimized. Then, use the query result of Q' to approximately answer Q. Note that it is possible that there are more than one aligned window minimizing $SM(Q, Q')$. In this case, choose the one whose area is the closest to Q. As depicted by Figure 11, both aligned windows $\{(1,0), (4,2)\}$ and $\{(0,0), (4,2)\}$ minimize SM; however $\{(1,0), (4,2)\}$ has the closest area to that of Q.

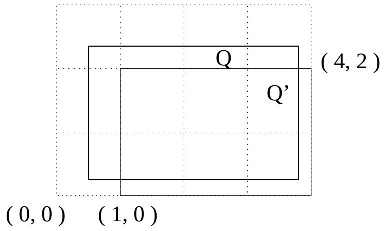


Fig. 11. Aligned Window Similar Object

Our algorithm is formally described in Algorithm 6.

Algorithm 6 Non-Aligned Window Query 1

Input:

non-aligned window query Q and the Euler histograms

Output:

$N_{cr}, N_{it}, N_{cs}, N_{cd}, N_{ds}$

Description:

Step 1. Obtain all Q's which are aligned windows and minimize $SM(Q, Q')$.

Step 2. Choose the Q' from Step 1 that minimizes the area difference between Q and Q'.

Step 3. Query Euler histograms by Q'.

Step 4. Return $N_{cr}, N_{it}, N_{cs}, N_{cd}, N_{ds}$ by Q' as the results for Q.

It should be clear that Algorithm 6 runs in constant time to get a Q'. This is because we need only to examine at most 16 neighbour aligned windows of Q.

5.2 Interpolation

For a query window Q, let Q_d denote the largest aligned rectangle that is contained by Q, and let Q_c denote the smallest aligned window that contains Q. Then we calculate the results for Q_d and Q_c, respectively, for a linear interpolation.

Suppose that l_1 and l_2 denote the distances between the two left vertical edges and the two right vertical edges in Q_d and Q_c, respectively. The distances between the two top horizontal edges and the two bottom edges in Q_d and Q_c are denoted by l_3 and l_4, respectively. We also denote the 4 corresponding distances between Q and Q_d by δ_1, δ_2, δ_3, and δ_4, respectively; see Figure 12 for illustration. Note that if Q is aligned then $Q_d = Q_c$ and $l_i = \delta_i = 0$ for $1 \leq i \leq 4$. Further, $0 \leq \delta_i \leq l_i$.

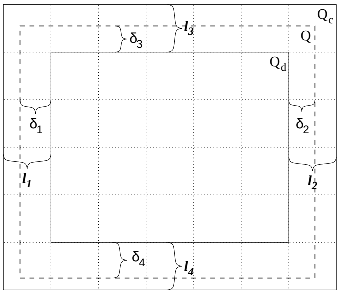


Fig. 12. Q, Q_d, and Q_c

Let N^c_{cs}, N^c_{ds}, and N^c_{cd} denote the results N_{cs}, N_{ds}, and N_{cd} regarding Q_c, respectively. The results N_{cs}, N_{ds}, and N_{cd} regarding Q_d are denoted by N^d_{cs}, N^d_{ds}, and N^d_{cd}, respectively, while the results N_{cs}, N_{ds}, and N_{cd} of Q are denoted by N^Q_{cs}, N^Q_{ds}, and N^Q_{cd}, respectively. Clearly,

$$\begin{aligned} N^d_{cs} \leq N^Q_{cs} \leq N^c_{cs} \\ N^c_{ds} \leq N^Q_{ds} \leq N^d_{ds} \\ N^c_{cd} \leq N^Q_{cd} \leq N^d_{cd} \end{aligned} \tag{11}$$

Based on the properties above, we use a simple liner interpolation to calculate N_{cs}^{Q}, N_{ds}^{Q}, and N_{cd}^{Q} as follows. Let $t = \frac{\sum_{i=1}^{4} \delta_i}{\sum_{i=1}^{4} l_i}$ and $t = 0$ if Q is aligned.

$$\begin{aligned} N_{cs}^{Q} &:= (1-t)N_{cs}^{d} + tN_{cs}^{c} \\ N_{ds}^{Q} &:= (1-t)N_{ds}^{d} + tN_{ds}^{c} \\ N_{cd}^{Q} &:= (1-t)N_{cd}^{d} + tN_{cd}^{c} \end{aligned} \tag{12}$$

Our techniques are summarized below in Algorithm 7.

Algorithm 7 Non-Aligned Window Query 2

Input:

non-aligned window query Q and the Euler histograms

Output:

$N_{op}, N_{cs}, N_{cd}, N_{ds}$

Description:

Step 1. Obtain Q_c and Q_d.

Step 2. Calculate N_{cs}^{d}, N_{cs}^{c}, N_{ds}^{d}, N_{ds}^{c}, N_{cd}^{d}, and N_{cd}^{c} with respect to Q_d and Q_c, respectively, according the techniques for aligned windows.

Step 3. Calculate N_{cs}^{Q}, N_{ds}^{Q}, and N_{cd}^{Q}, respectively, according to (12).

Step 4. $N_{op}^{Q} := |S| - N_{cs}^{Q} - N_{ds}^{Q} - N_{cd}^{Q}$.

Note that Algorithm 7 runs in constant time to get Q_c and Q_d, while the other costs are the same as those to query the histograms for aligned windows. If Q_d does not exist, then this algorithm uses the summarization results against Q_c as the results against Q.

6. UNEVEN DIVISION OF A DATA SPACE

In this section, we investigate the problem of dividing a data space unevenly to effectively support non-aligned window queries. We first present our results for minimizing the total "non-aligness" by unevenly dividing space. Then, we present histogram construction techniques and query processing techniques regarding an unevenly partitioned space.

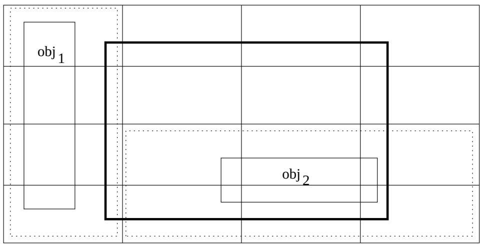


Fig. 13. Problems with Non-aligned Objects

6.1 Minimizing Non-aligness

Given a resolution $n_1 \times n_2$, a non-aligned object (i.e., not align with the grid $n_1 \times n_2$) is interpreted as the minimum aligned rectangle obj' that contains obj in an Euler histogram. Such approximation leads to errors of data summarization against a non-aligned window query.

As depicted in Figure 13, none of the two objects aligns with the partitioning lines in the grid. According to the Euler histogram construction algorithm, obj_1 is recorded in the histogram as the dotted-line rectangle containing obj_1; consequently, obj_1 intersects the query window (thick-line rectangle) in the histogram information while it actually disjoints with the query window. Such wrong topological relationship is also induced by obj_2. Moreover, an approximation of query windows makes the things more complicated.

It is immediate when all windows align with the grid or all objects align with the grid, such wrong topological relationships as shown above will not occur. This is why we do not investigate this issue in sections 3 and 4 while processing aligned windows.

To process non-aligned window queries (possibly ad-hoc), we need to make all objects be aligned with a grid to avoid such wrong topological relationships as above. This can be immediately done by inserting more partitioning lines into a given grid to make all objects aligned. However, this will cause a very large space requirement in case if most objects do not align with any of partitioning lines in a given grid.

To control the space usage, suppose that there are only l_1 vertical partitioning lines allowed and l_2 horizontal partitioning lines in the final space division. Moreover, we assume that the aligned window queries have to be supported regarding a given resolution $n_1 \times n_2$. Thus, we are allowed only to insert $k_1 = l_1 - n_1 - 1$ new vertical lines and $k_2 = l_2 - n_2 - 1$ new horizontal lines.

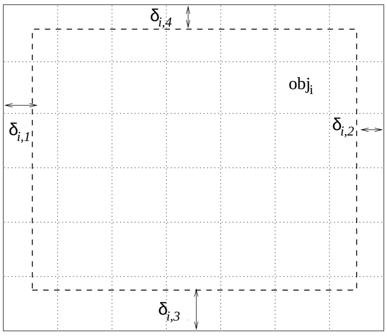


Fig. 14. An Illustration

As depicted in Figure 14, for each object obj_i, $\delta_{i,1}$ denotes the distance between the left boundary of obj_i and the left boundary of the minimal aligned rectangle Q_i containing obj_i, $\delta_{i,2}$ denotes the distance between the right boundary of obj_i and the right boundary of Q_i, $\delta_{i,3}$ denotes the distance between the bottom boundary of obj_i and the bottom boundary of Q_i, and $\delta_{i,4}$ denotes the distance between the top boundary of obj_i and the top boundary of Q_i. As obj_i is recorded in an Euler histogram as if obj_i fully occupied Q_i, we aim to insert k_1 and k_2 lines such that

(13) (i.e., the total non-aligness) is minimized, where n is the number of objects.

$$\sum_{i=1}^{n}(\delta_{i,1} + \delta_{i,2} + \delta_{i,3} + \delta_{i,4}) \tag{13}$$

Note that inserting k_1 vertical lines to minimize $\sum_{i=1}^{n}(\delta_{i,1} + \delta_{i,2})$ is independent of minimizing $\sum_{i=1}^{n}(\delta_{i,3} + \delta_{i,4})$ by optimally inserting k_2 horizontal lines. Below we present our algorithms to minimize $\sum_{i=1}^{n}(\delta_{i,1} + \delta_{i,2})$, which are based on "projecting" the problem on x-axis as follows. The minimization of $\sum_{i=1}^{n}(\delta_{i,3} + \delta_{i,4})$ can be similarly done by "projecting" the problem on y-axis and our algorithms for dealing with x-axis may be immediately applicable.

Suppose that we project the vertical partitioning lines in a given grid onto the points, called *projected grid points*, in the x-axis. The left and right boundaries of objects are also projected onto the points in the x-axis, called *projected object left-points* and *projected object right-points*, respectively. We want to find k_1 points in the x-axis so that the cost function in (14) is minimized. Such k_1 points correspond to additional k_1 vertical partitioning lines (i.e., each of these partitioning lines goes through one of such k_1 points), and are called *new partitioning points*.

$$\sum_{i=1}^{n}(\delta_{i,1} + \delta_{i,2}) \tag{14}$$

Note that the problem of *k-medians on a line* [Fleischer et al. 2004; Hassin and Tamir 1991] is similar to our problem; however the speed-up dynamic programming techniques (using time $O(k \log n)$) [Fleischer et al. 2004; Hassin and Tamir 1991] are not applicable to our problem due to the inherent difference between our problem and the problem of k-medians on a line. The following lemma is immediate.

Lemma 6.1. *The problem of selecting k_1 points to minimize (14) is equivalent to the problem of selecting k_1 projected object points to minimize (14).*

Proof. Suppose that x is a point selected in the optimal solution and x is not a projected object point. Suppose that a is the right-most projected object point but on the left side of x, and b is the left-most projected object point but on the right side of x. According to the definition of those δ values in (14), replacing x by the one from a and b, which leads to a smaller value of (14) between a and b, will never increase the value of (14). □

Without loss of generality, we assume that no projected object points are at the same positions as any projected grid points. Those projected object points at the same positions as some grid points can be eliminated for a further consideration since their δ values are already zero. Moreover, we may record one point in the x-axis ϕ times if ϕ object boundary lines are projected onto this point. We label the object points from 1 to n' according to the increasing ordering of their values where $n' \leq 2n$ and n is the number of objects.

A Dynamic Programming Based Algorithm. For $1 \leq i \leq j \leq n'$, let $M(i, j)$ denote the summation of the δ values of the projected object points from i to j if we select the ith projected object point and the jth projected object point, respectively, as two new partitioning points but no other new partitioning points between i to j. We use $M(0, j)$ to denote the situation where only the jth point among the first j object points is chosen as the new partitioning point while the left-most grid point is used as another partitioning point. $M(i, n'+1)$ denotes the situation where only the ith point is chosen as the new partitioning point among the object points from i to n' while the right-most grid point is used as another partitioning point. Note that when we calculate $M(i, j)$, we still should count the originally given grid partitioning points; $M(i, j)$ can be done in linear time. Let $M^*(j, l)$ denote the minimum value of (14) for optimally choosing at most l object points from the first j object points and the jth object point is used as a new partitioning point. Below is the crucial formula. For $1 \leq j \leq n'+1$,

$$M^*(j, l) = \min_{1 \leq i < j} \{M^*(i, l-1) + M(i, j)\} \tag{15}$$

Applying the dynamic programming technique in [Jagadish et al. 1998], the optimal solution $M^*(n'+1, k_1+1)$, minimizing (14) by choosing k_1 projected object points, can be calculated in time $O(k_1(n')^2)$. When n' is large, say, 1 million, the algorithm is too slow to finish though it is a polynomial time algorithm.

A Greedy Heuristic. We present an efficient algorithm based on a greedy paradigm. The basic idea is to iteratively choose a projected object point to greedily reduce the value of (14); the algorithm is outlined in Algorithm 8.

Algorithm 8 Greedy Data Space Partitioning

Input:

Projected grid points, object left-points, and object right-points.

Output:

New partitioning points

Description:

Iteratively choose a remaining object point to minimize (14) till k_1 iterations.

The algorithm trivially run in time $O(k_1 n')$. Below we present a more efficient way to execute each iteration of the algorithm. Note that although Algorithm 8 does not guarantee the optimality regarding minimizing (14), our performance evaluation demonstrates that it is very effective to serve our purpose for constructing a good histogram.

Speed-up. In each iteration, let x_m and x_M be a pair of two consecutive partitioning points (original or generated in earlier iterations), and let $x_m < x_M$. We denote all projected object left-points between x_m and x_M by $\tau = \{\tau_i : 1 \leq i \leq m_1\}$ and denote all projected object right-points between x_m and x_M by $r = \{r_i : 1 \leq i \leq m_2\}$. Note that both τ and r are sorted bags according to the increasing ordering. Suppose that a point x between x_m and x_M is chosen as a new partitioning point in this iteration. While the δ values of the other object points will not be changed, the δ values for points in τ and r will be changed. Specifically, it may be immediately verified the summation of these δ values, denoted by σ_{x_m,x,x_M}, is as follows if $\tau_j < x \leq \tau_{j+1}$, $r_{j'} \leq x < r_{j'+1}$, and x is chosen as a new partitioning point.

$$\sigma_{x_m,x,x_M} = [\sum_{i=1}^{m_1} \tau_i - jx_m - (m_1 - j)x] + [(m_2 - j')x_M - \sum_{i=1}^{m_2} r_i + j'x] \qquad (16)$$

Note that in (16) if we treat x before τ_1 as $j = 0$ and after τ_{m_1} as $j = m_1$, the formula still holds; similar treatment applies to x and $r_{j'}$. Since $(j' - m_1 + j)x$ is monotonic in $[r_{j'}, r_{j'+1}) \cap (\tau_j, \tau_{j+1}]$, we can immediately verify that only the object points are the candidates of such x to greedily reduce the value of (14). Moreover, such a cost function σ_{x_m,x,x_M} can be calculated by one scan of τ and r for all object points in τ and r. We can maintain a min-heap on all object points based on their cost functions σ and pop-up the heap every time. After pop-up the heap, we update the heap and the cost functions. To speed-up the computation, for each "bucket" (i.e., objects in two consecutive partitioning points) we build a min-heap and then build a min-heap on the bucket heaps. The time complexity of each iteration in Algorithm 8 is $O((m_1 + m_2)\log(m_1 + m_2) + \log(n_1 + k_1))$; recall $(n_1 + k_1)$ is the number of buckets in the final space devision.

6.2 Histogram Construction and Query Processing Techniques

With the data partitioning technique above for supporting non-aligned query windows, the data space division now is not necessarily even. Below we describe how we build and query an Euler histogram for an uneven data space partition.

Histogram Construction. The construction of an Euler histogram may be processed in the same way as that for even data space partition in section 2.5 using prefix-sum.

Query Processing Technique. As described in section 5, a non-aligned window is interpreted as a similar aligned window or an interpolation of two aligned windows. Now, we present how an aligned window is queried. This involves 1) calculating P_i and P_e, and 2) applying Algorithm 5.

P_i and P_e can be calculated using (6) and (7); however, $O(\log n_1 + \log n_2)$ time complexity is required to allocate the window to the histogram instead of constant time for evenly partitioned space.

To run Algorithm 5 in section 4.2, we need to maintain the information of scales. We could use exact scale (exact widths and heights) for each object; however this will lead to a huge number of scales - in the worst case, it equals the number of objects. To conform with the resolution in the histogram, a horizontal (or vertical) line from a to b has the width (or height) $d' - c$ if a falls in $[c, c']$ and b falls in $[d, d']$ according to the histogram partitioning. As depicted in Figure 15(a), the bottom line has the width 8 ($= 9 - 1$), while the top line's width is 5. Clearly, in the worst case there are $(n_1 + k_1) \times (n_2 + k_2)$ scales though in practice, the number of the scales could be much less than $(n_1 + k_1) \times (n_2 + k_2)$. With these scales specified, we can run Algorithm 5 against the scales and the windows where we replace the number of grid points by areas because the division is uneven.

Put Things Together. Suppose that we need to support aligned windows for a given resolution $n_1 \times n_2$ and we also have to support non-aligned windows with resolution at least $n_1 \times n_2$ (i.e. a non-aligned window contains at least a grid cell in the grid $n_1 \times n_2$.) Further suppose that we are allowed to build $(k + 1)$ Euler

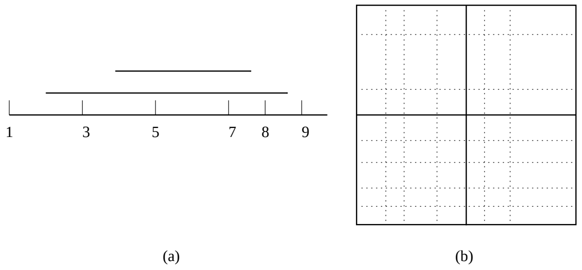


Fig. 15. An Example

histograms such that the first k Euler histograms have resolution $n_1 \times n_2$, while the last histogram could be inserted k_1 vertical lines and k_2 horizontal lines to the grid $n_1 \times n_2$. There are two ways to build these $(k+1)$ histograms:

(a). Use Algorithm 3 to divide objects into $(k+1)$ groups against the resolution $n_1 \times n_2$ such that for each of first k groups, build an Euler histogram regarding the resolution $n_1 \times n_2$. Apply Algorithm 8 to the $(k+1)$th group of objects regarding $n_1 \times n_2$, k_1 and k_2. Then, build the last histogram against the obtained uneven partitioning for this group of objects.

(b). Again, use Algorithm 3 to divide objects into $(k+1)$ groups against the resolution $n_1 \times n_2$ such that for each of first k groups, build an Euler histogram regarding the resolution $n_1 \times n_2$. Apply Algorithm 8 to the whole set of objects regarding $n_1 \times n_2$, k_1 and k_2. Then, build the last histogram against the obtained uneven partitioning for the whole set of objects.

Consider a multiscale histogram built by (a) or (b) has to support aligned window queries, as well as (ad-hoc) non-aligned window queries. We always use multiple histograms to process aligned window queries.

Querying an aligned window against the resolution $n_1 \times n_2$ regarding (a) may be executed in the same way as described in section 2.5 and section 4.2. Note that although in (b) we record the last histogram information for the whole data set, P_i and P_e regarding an aligned window with respect to the resolution $n_1 \times n_2$ (see the solid lines in Figure 15(b), for example) for the $(k+1)$th group of objects can be recovered by 1) calculating P_i and P_e from the $(k+1)$th histogram, and then 2) subtract the corresponding P_is and P_es in the first k histograms. Apart from this additional step, an aligned window against the resolution $n_1 \times n_2$ regarding (b) is executed exactly the same way as that in (a).

In (a) we keep the scales information for the $(k+1)$th group of objects regarding the obtained uneven space division, while in (b) we record such scales information for the $(k+1)$th group as well as the scales information for the whole set of objects. Consequently, for an aligned window against the resolution $n_1 \times n_2$ we run Algorithm 5 with respect to the scale information built on the $(k+1)$th group of objects. We process a non-aligned window in the following ways:

—Regarding (a), find the similar aligned window (or do interpolation of two aligned windows) against the resolution $n_1 \times n_2$ for the first k histograms and then obtain the exact summarization results for the aligned window (or windows). Find the similar aligned window (or do interpolation of two aligned windows) against the finer resolution for the last histogram, and then query the last histogram, as described above, regarding these approximate windows.

—Regarding (b), find the similar aligned window (or do interpolation of two aligned windows) against the finer resolution obtained for the last histogram, and then query the last histogram, as described above, regarding these approximate windows. Note that we need only query the last histogram as it contains the information for the whole set of objects.

Intuitively, (a) does not support our space partitioning technique, Algorithm 8, well if the last histogram contains very few objects; that is, the summarization accuracy will remain almost the same regardless how many new partitioning lines are added into the last histogram. Our performance study in section 9 confirms this.

7. SUMMARIZING SPATIAL DATASET IN 3D

Our techniques in a 2-dimensional space can be immediately extended to a 3-dimensional space for summarizing these four level-two topological relations. Below is a variation [Beigel and Tanin 1998] of Euler theorem [Harary 1969] in a d-dimensional space.

THEOREM 7.1. *If P is a bounded, connected d-dimensional polytope, then*

$$\sum_{0 \leq j \leq d} (-1)^{d-j} f_j(P) = 1. \tag{17}$$

Here, for $0 \leq j \leq d$, $f_j(P)$ denotes the number of j-dimensional interior faces of P.

Observing Theorem 7.1, Beigel and Tanin [Beigel and Tanin 1998] proposed to construct Euler Histogram for objects in a d-dimensional space. Restricted to a $3D$ space, the whole data space in $3D$-space is equally divided into $n_1 \times n_2 \times n_3$ cells. Then a bucket is allocated for each internal node, edge, face, and cell, such that:

—The integer, corresponding to a cell, is increased by 1 if an object intersects the cell.

—The integer, corresponding to an internal edge, is increased by 1 if an object crosses the edge.

—The integer, corresponding to an internal face, is decreased by 1 if an object passes the face.

—The integer, corresponding to a node, is decreased by 1 if an object contains the node.

Recall that P_i denotes the summation of all the bucket values inside an aligned query window Q (excluding the boundary). Theorem 7.1 immediately implies that $P_i = N_{nds}$ where N_{nds} is the number of objects non-disjointing Q. Thus, Euler Histogram can ensure exact solution of summarizing the level 1 relations for aligned windows in $3D$ space.

As with $2D$ space, the level 2 topological relations in $3D$ spaces are limited to four [Egenhofer 1989] after removing the boundary meeting conditions. This is

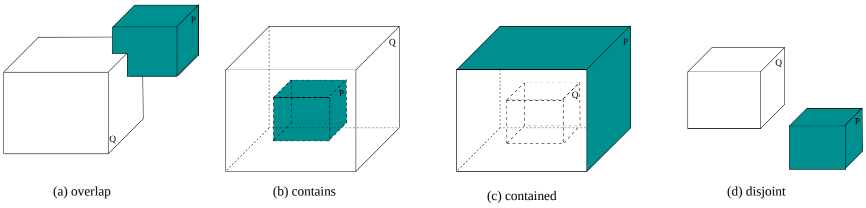


Fig. 16. Level-two Topological Relations in 3D

to break-down the non-disjointing relation into *contains* (cs), *contained* (cd), and *overlap* (op) (see Figure 16).

Similarly to the situation in a $2D$ space, the overlap relation in a $3D$ space should be divided into several classes as these classes contribute differently to the outside of Q: 1) *intersect* (Figure 17(a)), 2) *cross-overA* (Figure 17(b)), and 3) *cross-overB* (Figure 17(c)). Note that Cross-overA relation and Cross-overB relation are dual to each other. By Theorem 7.1 (a variation of Euler Theorem), the following theorem can be immediately verified.

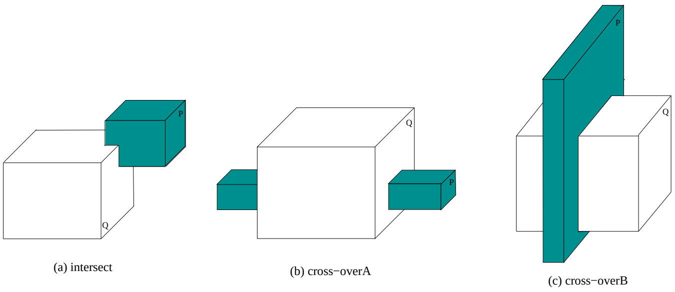


Fig. 17. Overlap Relations

THEOREM 7.2. *Suppose that P is an object and Q is an aligned window. Then, P contributes to summation of outside P_e of Q*

—*by* 1 *if the relation is intersect or disjoint,*

—*by* 2 *if the relation is cross-overA or contained.*

—*by* 0 *if the relation is cross-overB or contains.*

Suppose that H is a $3D$ Euler histogram for a set S of objects, and Q is an aligned window query. Let

—N_{crA} denote the number of objects in S with cross-overA relations to Q.

—N_{crB} denote the number of objects in S with cross-overB relations to Q.

—N_{it}, N_{cs}, N_{cd}, and N_{ds} are defined in the same way as those in $2D$ spaces (section 2).

Based on Theorem 7.1 and Theorem 7.2, we have:

$$|S| = N_{it} + N_{crA} + N_{crB} + N_{cs} + N_{cd} + N_{ds} \qquad (18)$$

$$P_i = N_{it} + N_{crA} + N_{crB} + N_{cs} + N_{cd} \quad (19)$$
$$P_e = N_{it} + 2 \times N_{crA} + 2 \times N_{cd} + N_{ds} \quad (20)$$

Now, there are 6 variables but only 3 equations. As with $2D$, the following theorem states that a histogram based on the objects with 8 "adjacent" scales in $3D$ space can guarantee the exact solutions.

THEOREM 7.3. *Suppose H is an Euler histogram with $n_1 \times n_2 \times n_3$ resolution. H can provide exact solutions to level-two relations in $3D$ space if objects involved in H with at most the 8 scales: (w, l, h), $(w+1, l, h)$, $(w, l+1, h)$, $(w+1, l+1, h)$, $(w, l, h+1)$, $(w+1, l, h+1)$, $(w, l+1, h+1)$, $(w+1, l+1, h+1)$.*

Similar arguments to those in the proof of Theorem 3.2 immediately lead to a proof of Theorem 7.3. Consequently, the algorithm MESA can be immediately extended to a $3D$ space to obtain an exact solution for aligned windows, while the 8 data sets on a $3D$ grid cell are used. Similarly, the problem to minimize the number of histograms in Algorithm MESA is NP-hard and the general version of *semi-local* optimization technique [Duh and Fürer 1997] can be applied as an approximate algorithm. In our implementation, we found that the semi-local optimization technique is also very effective in our 3D techniques.

If only $k + 1$ histograms are allowed, the $3D$ version of the problem WkP is also NP-hard for the same reason as that in $2D$ space. The algorithm GreedyWkP can be immediately extended to $3D$ space with the approximation ratio bounded by $1 - \frac{1}{e}$; this is because our proof of Theorem 4.2 does not depend on the dimensions.

Moreover, the techniques Algorithm Prob to summarize the last histogram can also be immediately extended to a $3D$ space based on the 8 cases in Theorem 7.3. Therefore, the algorithm MAPA has been extended to a $3D$ space. Finally, it should be immediate that the two algorithms for querying non-aligned windows, as well as our data space partitioning algorithms may be trivially extended to a 3D space.

8. MAINTENANCE AND APPLICATIONS

The Euler histograms generated by our techniques may be maintained as follows upon updates of dataset. Below, we use 2D space as an example; 3D datasets can be updated similarly. For an insertion, we need only to update the corresponding histogram (if more than one histogram involved) for the relevant node and cell values, respectively, by increasing 1; and update the relevant edge values, respectively, by decreasing 1. For a deletion, the updates to the corresponding histogram are opposite to those for an insertion. Further, if an insertion or deletion is involved in the last histogram, we need also to update their corresponding statistic values accordingly. Note that as the histogram values and the statistic information are stored in a cumulative fashion, an update needs to be propagated in a cumulative fashion as well. Moreover, we can also keep a threshold for the number of updates to trigger MAPA and our space partitioning algorithms (in section 6) to generate a new set of histograms.

It should be mentioned that the Euler histogram techniques are not only applicable to estimating spatial range query results but may also be immediately applicable to spatial digital libraries, as discussed earlier, to support window browsers [Beigel and Tanin 1998; Greene et al. 1999; Sun et al. 2002a; Szalay et al. 2000]. Further, our results are also fundamental to the development of new selectivity estimation techniques in spatial joins with the join predicates, such as contains, intersection, cross-over, etc.

9. PERFORMANCE EVALUATION

In this section we evaluate the performance of our new histogram techniques developed in this paper; Table I provides a summary. We first evaluate the techniques in 2 dimensional space. Then, we present our performance study for techniques in 3 dimensional space. All the experiments were conducted on Pentium IV 1.80GHz CPU with 512 Mbytes memory.

2D	Histogram Construction	MESA	Algorithm 1
		MAPA	Algorithms 3 & 4
	Aligned Window Queries	Prob, etc.	Algorithm 5 & Techniques in Section 2.5
	Non-Aligned Window Techniques	NAQ1	Algorithm 6
		NAQ2	Algorithm 7
		GDSP	Algorithm 8
		Prob_uneven	Described in Section 6.2
3D	Similar classification as 2D; All techniques in 2D are extendible to 3D	Similar notation as 2D but add 3D as a postfix	Described in Section 7

Table I. New Techniques

9.1 Techniques Evaluated in 2D Space

As the only technique [Sun et al. 2002a] dealing with the level-two topological relations in the 2 dimensional space for aligned window query, M-Euler will be used as a benchmark algorithm in the evaluation. Specifically, we evaluate the following techniques:

- M-Euler: Multi-resolution Euler Histogram techniques in [Sun et al. 2002a]. When only one histogram is allowed, it is the EulerApprox. It is for aligned windows.
- MAPA-Prob: Algorithm Prob is used to query the last histogram generated by MAPA, together with the techniques in section 2.5 to query the other histograms. It is for aligned windows.
- NAQ1: Similar window technique (Algorithm 6).
- NAQ2: Interpolation technique (Algorithm 7).
- MAPA-unevenA: MAPA is used to generate $(k + 1)$ histograms and GDSP is used to unevenly divide the grid for the $(k+1)$th histogram for a finer resolution. Prob_uneven and NAQ1 (or NAQ2) are used to query the last histogram, while NAQ1 (or NAQ2) is used to query the first k histograms. Note that NAQ1 (or NAQ2) runs against two different grids - one for the first k histograms and another for the last histogram.
- MAPA-unevenB: MAPA is used to generate $(k+1)$ histograms and GDSP is used to unevenly divide the grid for the $(k+1)$th histogram for a finer resolution. Here,

the last histogram is used for representing the whole set of objects as described in section 6. Prob_uneven and NAQ1 (or NAQ2) are used to query the last histogram. Although a non-aligned window query is processed against the last histogram, aligned window queries regarding the original resolution are processed against the $(k + 1)$ Euler histograms in the same way as MAPA-Prob (discussed in section 6.2).

We also evaluated the query time in these techniques as well as the histogram construction costs. Note that we evaluate the performance of MESA only against real datasets, since synthetic datasets used in the paper are designed to show the disadvantages of MESA - the necessity of developing MAPA; that is, there are a relatively large number of object scales.

9.2 Evaluating Aligned Window Queries in 2D Space

Datasets and Resolutions. In our experiment, real-world and synthetic datasets are used. To do a fair comparison with M-Euler regarding accuracy, we first adopt the 360×180 resolution to evaluate the accuracy of our algorithms against aligned windows, since this resolution was used in [Sun et al. 2002a] to provide experiment results. The 360×180 grid is a simulation of the earth resolution by the longitude and latitude. Then, we will also evaluate different resolutions to deliver a comprehensive performance study. Below are the datasets used.

—**Ca_road** consists of the $2,851,627$ California road segments obtained from the US Census TIGER [TIGER 2000] dataset. We normalized the dataset into the 360×180 grid; that is the space $[1, 360] \times [1, 180]$. For each road segment, its MBR is used.

—**Ca_Tx_road** consists of the $3,653,571$ Texas road segments (**Tx_road**) and the $2,851,627$ California road segments extracted from the US Census TIGER [TIGER 2000] dataset. We combine them together by normalizing both of them into the 360×180 grid. By combining the two real world datasets together, we hope that N_{cd} and N_{cr} may be reasonably significant. For each road segment, its MBR is used.

—**SAME** is a synthetic dataset used in [Sun et al. 2002a] such that each object has width 3.6 and height 1.8 in the space $[1, 360] \times [1, 180]$, while object positions follow a Zipf distribution [Zipf 1949]. This dataset is believed a simulation of many real world datasets.

—**Wet_USA_UT** is a combination of two real datasets, Wetland and USA_UT. The dataset Wetland contains $146,697$ wetland rectangular objects (MBR) from wetland map of US National Wetlands Inventory (www.uwi.fws.gov), while the dataset USA_UT contains $824,585$ rectangular stream objects in UT extracted from SGID&GIS data provided by Utah Automated Geographic Reference Center (agrc.utah.gov/agrc_sgid/sgidlib/statewide_gdb.htm). To merge them together, we normalize both of them into the data space $[1, 360] \times [1, 180]$. This set of objects contains a relatively large number of scales; it has over 200 different scales when the resolution is 210×105. It will be used to evaluate our MAPA techniques.

—**Zipf1** is a synthetic dataset with one million square objects. The centers of the objects are uniformly distributed over the 360 × 180 grid, while the side lengths follow a Zipf distribution and objects are all aligned with the grid.

—**Zipf2** is to add $250,000$ objects with the scale $(1, 50)$ and $250,000$ objects with the scale $(30, 30)$ on the top of Zipf1 dataset. This dataset will produce large values of N_{cr} and N_{cd}. Though it is quite unusual in real world, it is expected to further confirm the advantages of our algorithms.

Query Sets. We select the query windows to accommodate variously different user query patterns. We divide query windows into two classes, small and non-small. A query window in small class has a scale such that the width and height are both smaller than 5, while a query window in non-small class has either height between 5 and 20 or width between 5 to 20. We randomly generate 3 different aligned window query sets, T_l, T_m, and T_s, each of which has $100,000$ query windows.

In T_l, 20% of the $100,000$ query windows are in the small class. In T_m, 40% of the query windows are in the small class, while in T_s, 80% of the query windows are in the small class. In each of these 3 datasets, small queries and non-small queries are all randomly generated regarding scales and locations, respectively.

Error Metrics. Let T be a set of query windows to be evaluated. For each $Q \in T$, we record the relative errors for N_{cs}, N_{cd} and N_{ov}, respectively, where $N_{ov} = N_{it} + N_{cr}$. Recall that the relations it and cr are subdivided from the relation ov, and we aim only to summarize the relation ov. The relative error is defined below.

$$\epsilon = \begin{cases} \frac{|e-e'|}{e} & \text{if } e \neq 0 \\ e' & \text{otherwise} \end{cases} \tag{21}$$

Here, e is the exact value and e' is an approximate value. The average relative error may be defined below.

$$\frac{\sum_{Q \in T} \epsilon_Q}{|T|} \tag{22}$$

Here, ϵ_Q is the relative error for a query window Q.

Effectiveness of MESA. In the dataset SAME used in [Sun et al. 2002a], the objects have 4 different scales with respect to the 360 × 180 resolution: $(4, 2)$, $(4, 3)$, $(5, 2)$, and $(5, 3)$. While M-Euler cannot provide the exact solutions even with 4 histograms, MESA can always guarantee the exact solutions by only one Euler histogram. Further, in this application the querying time of MESA is about 4 times less than M-Euler when M-Euler uses 4 histograms. We will present the experiment results regarding the histogram construction time and querying time later together with the other algorithms.

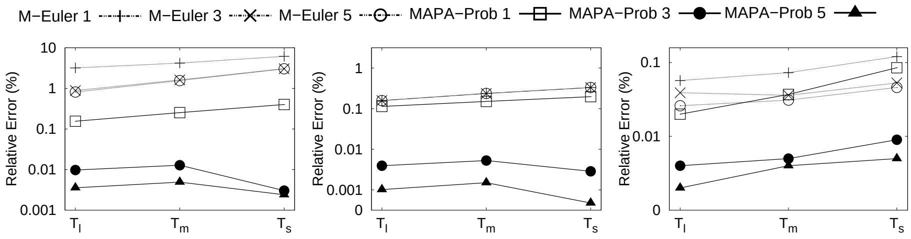
We examined the number of histograms produced by MESA, respectively, against the 3 resolutions 100 × 50, 180 × 90, and 360 × 180 for Ca_road and Tx_road. The numbers of histograms generated in Ca_road are 3, 8, and 17, respectively, while the breakdown numbers for Tx_road are 4, 9, and 13, respectively. Consequently, MESA is practical (i.e., requires small number of histograms) against Ca_road and

Tx_road.

Note that the dataset Wet_USA_UT requires over 100 histograms for the resolution 360 × 180; this is because over 300 scales are presented. Therefore, for such datasets the MAPA technique is necessary.

Accuracy of MAPA. We examine the approximation accuracy of the algorithms M-Euler and MAPA-Prob, against 3 different storage space requirements: 1 histogram, 3 histograms, and 5 histograms. In our experiments, we examine only the accuracies of N_{cs}, N_{cd}, and N_{ov} but N_{ds} is omitted; this is because these 2 algorithms are always able to produce the exact answers to N_{ds} (see equation (2)). We recorded the average relative errors for a given storage space and a given query set for these 2 algorithms, respectively.

Our algorithm MAPA automatically generates a set of histograms but in M-Euler we need to intuitively, manually specify the data partitioning to obtain a good set of histograms.

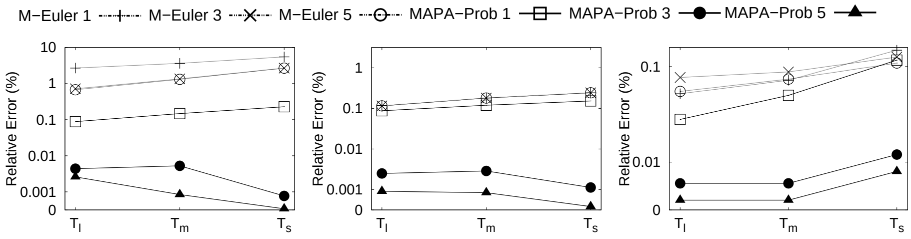


(a) Ncs (b) Nov (c) Ncd

Fig. 18. California Road Segments Data

In the first set of experiments, we do the performance evaluation regarding the environment in [Sun et al. 2002a]; that is, the given resolution is 360 × 180.

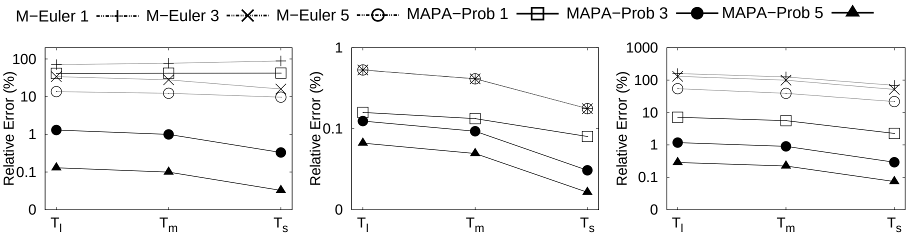
Figure 18 shows the experiment results against Ca_road, where M-Euler 1, 3, 5 denote the experiment results of M-Euler while using 1, 3, 5 histograms, respectively. Similar notation is also applied to MAPA-Prob. The experiment results demonstrated that MAPA-Prob greatly improve the accuracy of M-Euler, while MAPA-Prob 5 may improve the accuracy of M-Euler by up to two orders of magnitude.



(a) Ncs (b) Nov (c) Ncd

Fig. 19. California & Texas Road Segments Data

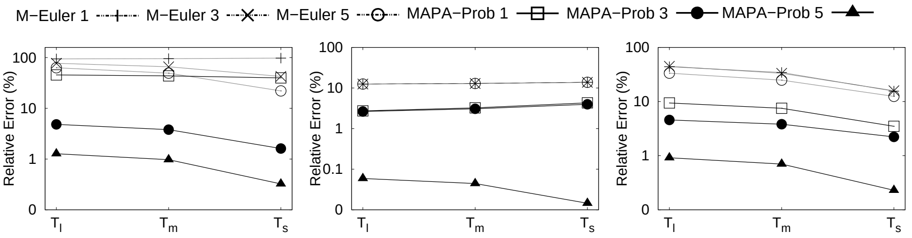
The experiment results for dataset Ca_Tx_road continue the trends, as depicted in Figure 19.



(a) Ncs (b) Nov (c) Ncd

Fig. 20. Synthetic Data – Zipf1

Figure 20 presents the experiment results for the synthetic dataset Zipf1. In our implementation of M-Euler, we use the data partitioning suggested in [Sun et al. 2002a]. To generate 3 histograms, the first histogram contains the objects with the areas 1 to 8, the second histogram contains the objects from areas 9 to 99, and the third histogram contains the objects with the areas 100 and more. To generate 5 histograms, the first histogram contains the object with the areas from 1 to 8, the second with the areas from 9 to 24, the third with the areas from 25 to 99, the fourth with the areas from 100 to 224, and the fifth with the areas 225 and more. The experiment results follow similar trends to those in Ca_road. It is interesting to note that MAPA-Prob 1 already greatly out-performs M-Euler 5 with respect to N_{ov} and N_{cd} though the storage space required by MAPA-Prob 1 is about 4 times smaller than that in M-Euler 5.



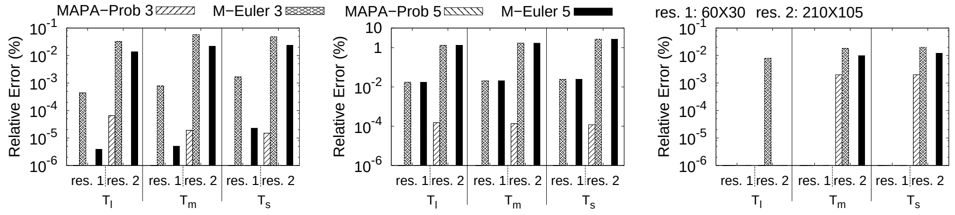
(a) Ncs (b) Nov (c) Ncd

Fig. 21. Synthetic Data – Zipf2

Similar trends to those in Zipf1 continue in Zipf2, as depicted in Figure 21. It is worth to note that the accuracy of estimating N_{ov} in M-Euler is always fixed regardless of the number of histograms to be used; this may be problematic as illustrated by Figure 21 (b).

In the second set of experiments, we evaluate our techniques against different resolutions from 60×30 to 210×105. For each resolution, we generate three query sets T_l, T_m, and T_s in the same way as described above. Here, we limit the number of queries in each dataset to be $50,000$ since the resolution 60×30 has only about $80,000$ small windows. Note that different resolutions lead to different T_l, T_m, and T_s.

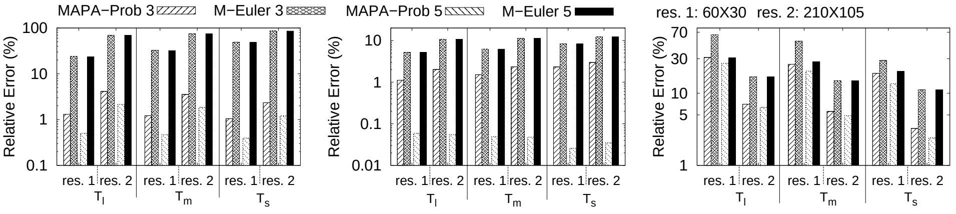
Clearly, a higher resolution Euler histogram gives more aligned windows. However, a higher resolution does not necessarily imply higher accuracy for aligned window queries. An extreme case is the lowest resolution 1 × 1 which has only one aligned window but the exact answers. In fact, aligned windows regarding two different resolutions are often different. Moreover, if two aligned windows regarding two different resolutions, respectively, happen to be the same, they will have the same total P_is and P_es; in this case, the number and the scale distribution of objects in the last histogram determine the accuracy. This set of experiments verify the above intuition.



(a) Ncs (b) Nov (c) Ncd

Fig. 22. Aligned Window Query: Ca_Tx_road Dataset

Figure 22 shows the experiment results against the dataset Ca_Tx_road regarding the resolutions 60 × 30 and 210 × 105. Clearly, our techniques still outperform M-Euler by several orders of magnitude in each different resolution. Note that in our experiments, several results achieve 0% relative error (i.e., provide exact answers); consequently, their error bars are not displayed. For this set of data, the lower resolution 60 × 30 leads to a slightly higher accuracy for aligned windows than that for the higher resolution 210 × 105 because only a few hundred objects left in the last histogram regarding the resolution 60 × 30.



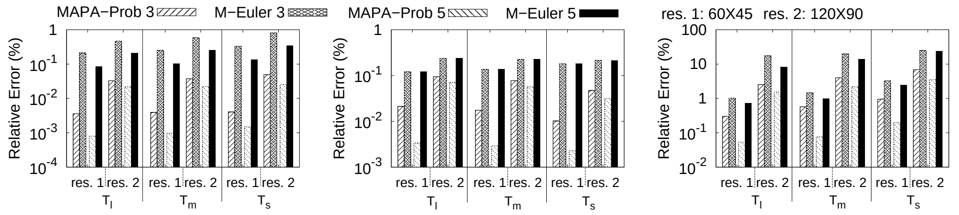
(a) Ncs (b) Nov (c) Ncd

Fig. 23. Aligned Window Query: Zipf2 Dataset

Figure 23 reports the results of our performance study against the dataset Zipf2 regarding the resolutions 60 × 30 and 210 × 105, respectively. It also shows that the significant improvement made by our technique over M-Euler. This experiment also demonstrates that a lower resolution does not warrant a higher accuracy for aligned window queries; in this dataset, the numbers of objects left in the last histogram regarding the two resolutions, respectively, do not have a big difference as above.

Figure 24 depicts our experiment results against the dataset Wet_USA_UT regarding the resolutions 60 × 45 and 120 × 90, respectively. It further validates the trends obtained in the last two experiments.

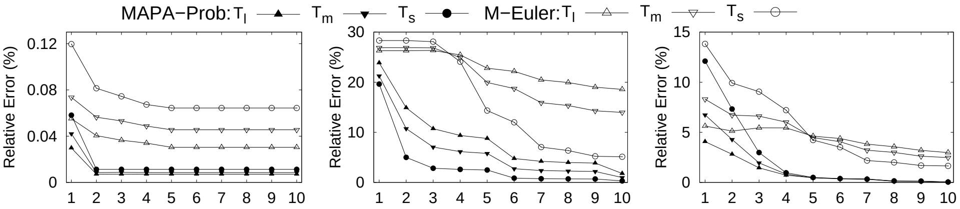
Accuracy vs k. In this set of experiments, we study the relationships between k and accuracy. The 3 datasets are used: Ca_Tx_road, Zipf1, and Wet_USA_UT. We



(a) Ncs (b) Nov (c) Ncd

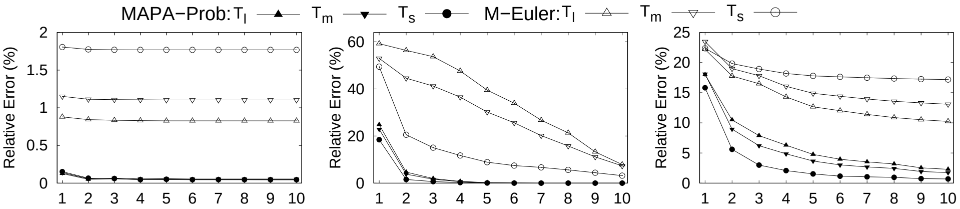
Fig. 24. Aligned Window Query: Wet_USA_UT Dataset

conduct the experiment against two resolutions: 180 × 90 and 360 × 180. For each resolution, 3 sets of aligned window queries, T_l, T_m, and T_s, are generated in the same way as described earlier. Each query set has 50,000 queries.



(a) Ca_Tx_road (b) Zipf1 (c) Wet_USA_UT

Fig. 25. Histograms for Different k (res. 180 × 90)



(a) Ca_Tx_road (b) Zipf1 (c) Wet_USA_UT

Fig. 26. Histograms for Different k (res. 360 × 180)

The experiment results are depicted in Figures 25 & 26. We record average relative errors by averaging relative errors regarding N_{cs}, N_{ov}, and N_{cd}, respectively. We evaluate 1 histogram to 10 histograms. They show that MAPA always leads to significantly smaller errors than those by the existing technique M-Euler.

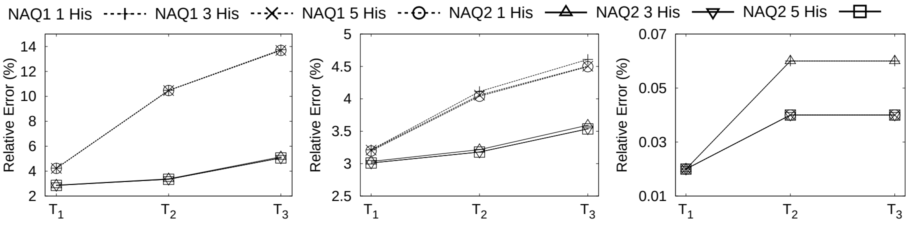
9.3 Evaluating Non-Aligned Window Queries

We evaluate our techniques only since there is no existing technique to process non-aligned window queries with respect level-two topological relations.

NAQ1 & NAQ2. In our first set of experiments, we evaluate NAQ1 and NAQ2 only and do not apply our space partitioning algorithm. We conduct experiments against the 360 × 180 resolution. The following data is used in addition to the real dataset Ca_road.

—**Zipf3** is a synthetic dataset with one million square objects. It is designed to test our non-aligned window querying techniques. The objects are not necessarily aligned to the grid and distributed to any positions over the 360×180 grid, while the side lengths follow a Zipf distribution. The maximum side length is designed by $\sqrt{180}$.

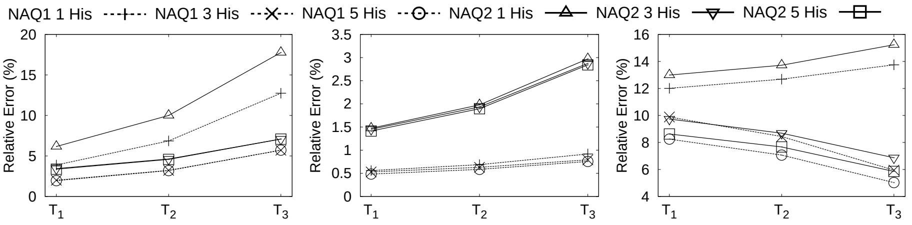
Three sets of window queries are generated, T_1, T_2, and T_3, each of which has $50,000$ query windows. To simulate real applications, in each query window set we generate 50% non-aligned window queries and 50% aligned window queries. The window size distributions in T_1, T_2, and T_3 are similar to T_l, T_m, and T_s, respectively.



(a) Ncs (b) Nov (c) Ncd

Fig. 27. Non-Aligned Window Queries: Ca_road Dataset

Figure 27 reports our experiment results for query sets T_1, T_2, and T_3 against the Ca_road dataset. It is interesting to notice that both algorithms are not very sensitive to the increment of the number of histograms when the number is increased to 3. While NAQ2 outperforms NAQ1 for N_{cs} and N_{ov}, NAQ1 and NAQ2 perform similarly regarding N_{cd} when the number of histograms is at least 3.



(a) Ncs (b) Nov (c) Ncd

Fig. 28. Non-Aligned Window Queries: Zipf3 Dataset

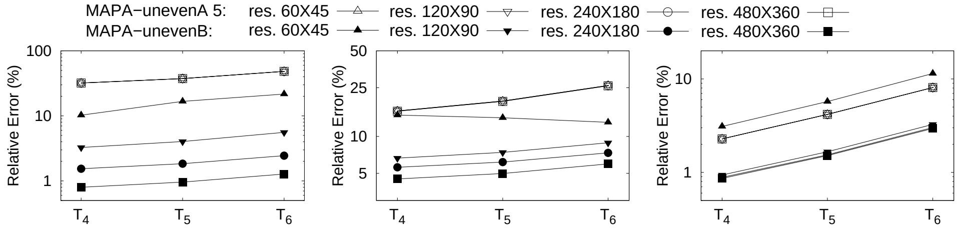
We also evaluate NAQ1 and NAQ2 against the Zipf3 dataset. This time an increment of the number of histograms does improve the accuracy especially for N_{cs} and N_{cd}. Interestingly, NAQ1 significantly outperforms NAQ2 regarding N_{cs} and N_{ov}, while they still perform similarly for N_{cd}.

Our experiment results demonstrated that both NAQ1 and NAQ2 are effective. NAQ1 is more accurate than NAQ2 for synthetic data, while NAQ2 gives an edge over NAQ1 for the real dataset used. In the remaining part of evaluating non-aligned windows, we use NAQ1 in MAPA-unevenA and MAPA-unevenB.

Grid Partitioning Algorithm. We evaluate all of our techniques for processing non-aligned windows; that is, MAPA-unevenA and MAPA-unevenB. The real dataset used is Wet_USA_UT that has a big number of scales; for example, it has 458 different scales when the resolution is 480 × 360. To challenge our data space partitioning techniques, the following synthetic dataset Zipf4 is generated with a large number of scales. We also give a Zipf distribution of the object locations in Zipf4.

—**Zipf4** is a synthetic dataset consisting of 3 millions rectangle objects which are allocated in the space $[1, 360] \times [1, 180]$. We generate 180 different scales as follows. For each scale, one side length a follows a Zipf (z=1) distribution for the integers in $[1, 180]$, while another side length b is randomly chosen from $[1, 180]$. The centers of objects follow a Zipf (z=0.5) distribution on the grid cells regarding the resolution 360 × 180, and the actual locations of the centers in a cell follow a random (uniform) distribution.

In the second set of experiments, we evaluate only non-aligned windows. We define three non-aligned windows T_4, T_5, and T_6 such that the percentages of small and non-small are the same as T_l, T_m, and T_s, respectively, as well as the locations and scales' distributions. To ensure each window covers at least one cell, a small window regarding a given resolution has the constraint that both height and width intersect 3 to 7 cells, respectively. A non-small window has one side intersecting 7 to 20 cells and another side intersecting 3 to 7 cells.

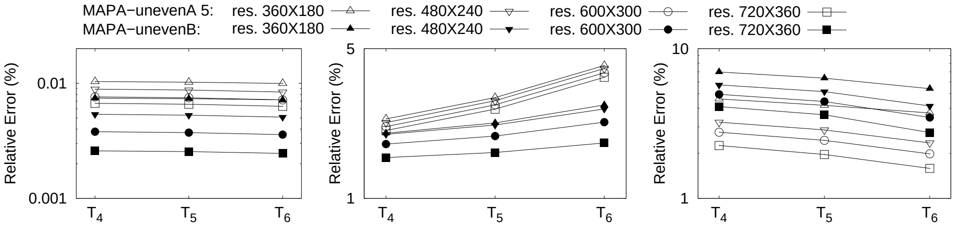


(a) Ncs (b) Nov (c) Ncd

Fig. 29. Non-Aligned Window Query: Wet_USA_UT Dataset

Figure 29 demonstrates our experiment results against the dataset Wet_USA_UT, and T_4, T_5, and T_6 specified on 60×45. In this experiment, we increase the resolution by inserting additional 60 vertical lines and 45 horizontal lines to make a resolution 120 × 90 using GDSP (Algorithm 8). Applying GDSP, the obtained resolution 120 × 90 is increased to 240 × 180; applying GDSP again, the resolution 240 × 180 is increased to 480 × 360. The experiment clearly demonstrates that in MAPA-unevenB, the accuracy is increased while the resolution is increased. However, MAPA-unevenA 5 does not share the trends. This is because in MAPA-unevenA 5, the number of objects is relatively small - only a few hundreds regarding a very low resolution 60 × 30 and we only increase the resolution in the last histogram, while the errors (the dominant part) in the first 4 histogram remain unchanged. In such a situation, MAPA-unevenB, dealing the whole set of objects by using one histogram, outperforms MAPA-unevenA 5.

To further validate the above observation, we do the experiment against Zipf4 starting from a relatively high resolution 360×180. Here, there are about 1, 917, 364 objects left in the last histogram of MAPA-unevenA 5. Figure 30 shows the exper-



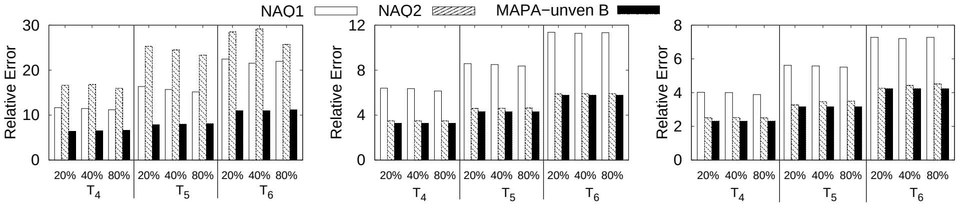
(a) Ncs (b) Nov (c) Ncd

Fig. 30. Non-Aligned Window Query: Zipf4 Dataset

iment results against the dataset Zipf4 and T_4-T_6 specified on 360×180. Instead of applying GDSP to increase the resolution 360×180 to 720×360 at once, we break down this into three stages from 360×180 to 480×240, then to 600×300, and then to 720×360. We evaluate our techniques against each stage. This experiment also demonstrates that in MAPA-unevenB a higher resolution leads to a higher accuracy for non-aligned window summarizations. However, as about two third objects left in the last histogram, in MAPA-unevenA 5 a higher resolution also leads to a higher accuracy. Moreover, MAPA-unevenB does not always outperforms MAPA-unevenA as MAPA-unevenA distributes the objects into another 4 histograms (e.g. MAPA-unevenA 5) and makes the use of the information in these 4 histograms.

As discussed and demonstrated in section 9.2, summarization against aligned windows regarding different resolutions does not imply a relationship between the accuracy and resolution. This is simply because aligned windows against two different resolutions are usually different. The above two experiments, however, showed a strong co-relation between a higher resolution and a higher accuracy for a same set of window queries (non-aligned windows in general).

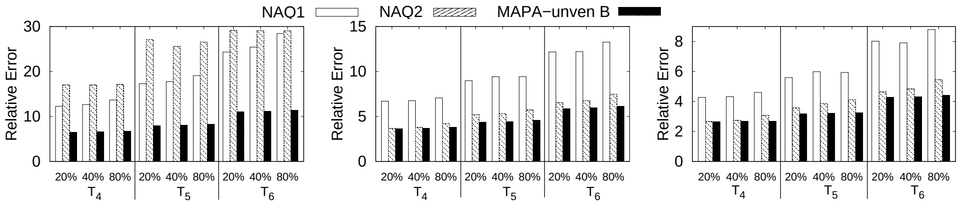
Finally, we evaluate the effectiveness of grid partitioning algorithms by examining whether or not it improves the accuracy by using equal partitions only. Since grid re-partitioning techniques are applied to the last histogram only, we focus on one histogram for such a performance evaluation; consequently, MAPA-unevenA degenerates into MAPA-unevenB.



(a) res. 60×45 (b) res. 120×90 (c) res. 240×180

Fig. 31. Wet_USA_UT Dataset for Queries Favouring Even Partitioning

We conduct the experiment against the real dataset Wet_USA_UT. The re-partitioning process starts from the resolution 30×30. We record the experiment results for resolutions 60×45, 120×90, and 240×180, with respect to three techniques NAQ1, NAQ2, and MAPA-unevenB. In MAPA-unevenB, our grid re-partitioning algorithms are used to get those 3 high resolutions, while in NAQ1 and NAQ2 equal grid partitions are used in combining with NAQ1 and NAQ2, respectively. We record the average relative errors over all estimations for N_{cs}, N_{ov}, and N_{cd}, respectively.



(a) res. 60 × 45 (b) res. 120 × 90 (c) res. 240 × 180

Fig. 32. Wet_USA_UT Dataset for Queries Favouring Non-even Partitioning

The 3 query sets T_4, T_5, T_6, defined on 60×45, are used with some perturbation towards grid partitioning lines. Each query set T_i (for $i = 4, 5, 6$) is perturbed into 3 query sets. In the first perturbed query set, 20% queries are randomly selected, so that each boundary line of a selected query window is assigned a random distance (not greater than 0.2) away from its closest partitioning grid line, respectively, if the original distance is greater than 0.2; for example, for the left boundary line of an object we always choose the right-most grid partitioning line but on the left side of this object line. In the second perturbed query set, 40% queries are randomly chosen to do the same perturbation, while in the third perturbed query set, 80% queries are chosen. For the experiment results depicted in Figure 31, such perturbation favours equal partitions; that is, partition lines are chosen from equal grid partitions. For the experiment results depicted in Figure 32, perturbations are against partition lines generated by our partitioning algorithms. The experiment results clearly demonstrate that MAPA-unevenB outperforms NAQ1 and NAQ2. However, when resolution increases, the accuracy differences between MAPA-unevenB and NAQ1 (or NAQ2) get smaller; this is because those queries become large window queries when the resolution increases.

9.4 Query Processing, Histogram Construction, and Update Costs

As analyzed earlier, the time for querying an Euler histogram in M-Euler and MAPA-Prob, is constant, respectively, which is irrelevant to the size of the Euler histogram and the underlying spatial data. Our experiment results (based on Ca_Tx_road) in Figure 33 confirmed this.

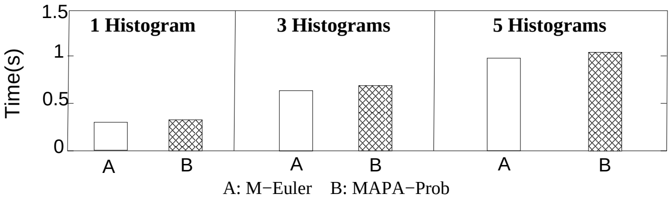


Fig. 33. Query Time

In fact, our implementation of these 2 algorithms against all the data demonstrated that the query time depends only on the number of histograms required and the types of the algorithm used. Note that the algorithm Prob is slightly slower than the algorithm EulerApprox, and the algorithm EulerApprox is slightly slower than the algorithm for solving the linear equations (2) - (4) directly when two variables are zero. These have been reflected in the experiment results.

We implemented MESA against Ca_Tx_road; it takes about 5 seconds for $100,000$ window queries; this is because there are 19 histograms involved.

We also evaluated the efficiency of NAQ1 and NAQ2. Both techniques need to calculate some aligned windows first to answer a query. As these aligned windows are calculated in constant time, the total costs for NAQ1 and NAQ2 are still constant, respectively. In fact, they are very close to those for MAPA-Prob.

Finally, we evaluate the query processing time of MAPA-unevenB against Zipf4 regarding T_4-T_6 against the resolution 480×360 obtained by further dividing 360×180 using GDSP. On average, each query is processed by 7.2×10^{-5} second. Note that the query processing time of MAPA-unevenA is about the total time of running MAPA-unevenB and MESA (for k histograms).

Histogram Construction and Update Time. We first evaluate the running time to construct the histograms against an evenly divided data space. In the datasets used in the experiments, Ca_Tx_road has the largest number of objects, about $6,500,000$. The time costs for constructing the histograms in M-Euler and MAPA-Prob, respectively, for 1 histogram, 3 histograms, and 5 histograms with the two resolutions, 360×180 and 180×90 are between 40 seconds and 41 seconds. In fact, the costs of these 3 algorithms, for constructing the histograms, are dominated by the costs of scanning the dataset; this is why those construction costs are similar.

We also recorded the histogram construction time in MESA against the data Ca_Tx_road. It takes about 49 seconds for the resolution 360×180, and about 42 seconds for the resolution 180×90. The construction time in MESA for 360×180 is significantly higher than those of M-Euler and MAPA-Prob due to the costs of a search for the right histogram for each object and the costs for computing the bucket values, as there are 19 histograms involved.

Finally, we record the total time of applying GDSP to divide the data space to increase a grid resolution 360×180 to 720×360 regarding the dataset Ca_Tx_road, and then building the histogram based on such a resolution 720×360 for the dataset Ca_Tx_road. The total time is 5.4 seconds among which 3 seconds are used to do an initial sorting on 6.5M objects in Algorithm 8; and 2 seconds are used to build the histogram. Only 0.4 seconds are used to do iterations in Algorithm 8.

We also record the average time for inserting $10,000$ random objects into the above histogram; it is 1.51×10^{-3} seconds. Note that the cost of deleting an object is similar to that of inserting.

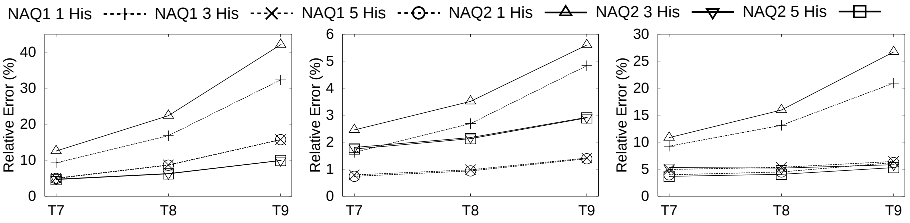
9.5 Performance Study of 3D Techniques

As the efficiencies of our techniques in 3D space are very similar to those for 2D space, In this subsection we present only our performance study results regarding approximation accuracy.

Dataset and Resolution. A $180 \times 180 \times 180$ grid resolution is used in our performance study. Below is the dataset used. **Zipf5** is a synthetic dataset with five million 3 dimensional hype-rectangular objects in 3D space. The object centers are random generated over the 180 × 180 × 180 grid, while the side lengths follow a Zipf distribution. The maximum side length is designed by $\sqrt{180}$.

Query Sets. Three query sets, T_7, T_8 and T_9, are generated, each of which has 50,000 objects. The distributions of window sizes in T_7, T_8, and T_9 are similar to those of T_4, T_5, and T_6, respectively. Moreover, as with 2D experiments in each of T_7, T_8, and T_9 there are 50% non-aligned window queries and 50% aligned window queries.

Error Metrics. The relative error metrics as described in the last subsection. That is, for each $Q \in T_i$ $(7 \leq i \leq 9)$, we record the relative errors for N_{cs}, N_{cd} and N_{ov}, respectively, where $N_{ov} = N_{it} + N_{crA} + N_{crB}$.



(a) Ncs (b) Nov (c) Ncd

Fig. 34. 3D Zipf5 Dataset

Approximation Accuracy. In our experiment, we evaluate the accuracy of NAQ1_3D, NAQ2_3D, MAPA_3D. Figure 34 gives our experiment results. Our experiment demonstrated that NAQ1 and NAQ2, in combining with MAPA_3D, are effective and can achieve accuracy not worse than 5% when 5 histograms are used.

9.6 Summary

Our performance study in this section showed that the techniques developed in this paper are not only highly accurate but also very efficient. Our techniques outperform the existing techniques (for aligned windows) in 2D space by several orders of magnitude regarding accuracy in most cases. This is because our multi-scale techniques effectively capture the properties of exact solutions, as well as the effectiveness of our new techniques for summarizing one histogram which does not guarantee the exact answers.

Moreover, our techniques MAPA-unevenA and MAPA-unevenB for processing non-aligned windows are both very efficient and effective (accurate). Our experiments demonstrate that in case that not many objects are left in the last histogram, MAPA-unevenB is a much better choice if we are allowed to increase the resolution.

Our experiments not only cover the applications with limited number of scales but also demonstrate high accuracy and efficiency against the possible applications where the number of scales is large. Note that our grid partitioning techniques do not address the query patterns. Consequently, for non-aligned window queries increasing resolution becomes the only alternative when errors are large (see Figure

29 for example).

10. CONCLUSION AND REMARKS

In this paper, we investigate the problem of effectively summarizing the level-two topological relations against large spatial datasets by histograms. By effectively utilising the object scale information, we first present an efficient algorithm MESA to construct a small set of histograms, based on a multi-scale paradigm, to provide exact summarization results for aligned windows. To conform to a limited storage space, we provide an effective algorithm MAPA to construct a fixed number of histograms with the aim to minimize the estimation errors. We also present a novel and effective approximate algorithm, Prob, to query one histogram; it runs in constant time. Then, we present novel techniques to process non-aligned windows Finally, we extend our techniques to 3D space. Our experiment results demonstrate that our techniques, developed in this paper, greatly improve the accuracy of the existing techniques while retaining the costs efficiency.

Electronic Appendix

The electronic appendix for this article may be accessed in the ACM Digital Library. The appendix contains the proof of Theorem 3.3.

REFERENCES

Aboulnaga, A. and Naughton, J. F. 2000. Accurate estimation of the cost of spatial selections. In *ICDE'00*. Washington, DC, February 28 - March 3, 123–134.

Acharya, S., Poosala, V., and Ramaswamy, S. 1999. Selectivity estimation in spatial databases. In *SIGMOD'99*. Philadelphia, Pennsylvania, June 1-3, 13–24.

Beigel, R. and Tanin, E. 1998. The geometry of browsing. In *LATIN'98*. Brazil, April 20-24, 331–340.

Duh, R.-C. and Fürer, M. 1997. Approximation of k-set cover by semi-local optimization. In *STOC'97*. El Paso, Texas, May 4-6, 256–264.

Egenhofer, M. J. 1989. A formal definition of binary topological relationships. In *FODO'89*. Paris, June 21-23, 457–472.

Egenhofer, M. J. and Herring, J. R. 1994. Categorizing binary topological relations between regions, lines, and points in geographic databases. In *The 9-Intersection: Formalism and Its Use for Natural-Language Spatial Predicates. National Center for Geographic Information and Analysis, Report 94-1*, M. J. Egenhofer, D. M. Mark, and J. R. Herring, Eds.

Feige, U. 1998. A threshold of ln n for approximating set cover. *JACM 45,* 4, 634–652.

Fleischer, R., Golin, M. J., and Zhang, Y. 2004. Online maintenance of k-medians and k-covers on a line. In *SWAT'04*. Humlebaek, Denmark, July 8-10, 102–113.

Gaede, V. and Günther, O. 1998. Multidimensional access methods. *ACM Computing Surveys 30,* 2, 170–231.

Garofalakis, M. N. and Gehrke, J. 2002. Querying and mining data streams: You only get one look. In *VLDB'02 Tutorial*. Hong Kong, August 20-23.

Gilbert, A. C., Kotidis, Y., Muthukrishnan, S., and Strauss, M. 2002. How to summarize the universe: Dynamic maintenance of quantiles. In *VLDB'02*. Hong Kong, August 20-23, 454–465.

Greene, S., Tanin, E., Plaisant, C., Shneiderman, B., Olsen, L., Major, G., and Johns, S. 1999. The end of zero-hit queries: Query previews for nasa's global change master directory. *International Journal on Digital Libraries 2,* 2-3, 79–90.

Grigni, M., Papadias, D., and Papadimitriou, C. H. 1995. Topological inference. In *IJCAI (1)*. Montréal, Québec, August 20-25, 901–907.

HADJIELEFTHERIOU, M., KOLLIOS, G., AND TSOTRAS, V. J. 2003. Performance evaluation of spatio-temporal selectivity estimation techniques. In *SSDBM'03*. Cambridge, MA, July 9-11, 202–211.

HARARY, F. 1969. *Graph Theory*. Addison-Wesley.

HASSIN, R. AND TAMIR, A. 1991. Improved complexity bounds for location problems on the real line. *Operations Research Letters 10*, 395–402.

HO, C.-T., AGRAWAL, R., MEGIDDO, N., AND SRIKANT, R. 1997. Range queries in olap data cubes. In *SIGMOD'97*. Tucson, Arizona, May 13-15, 73–88.

JAGADISH, H. V., KOUDAS, N., MUTHUKRISHNAN, S., POOSALA, V., SEVCIK, K. C., AND SUEL, T. 1998. Optimal histograms with quality guarantees. In *VLDB'98*. New York City, New York, August 24-27, 275–286.

JIN, J., AN, N., AND SIVASUBRAMANIAM, A. 2000. Analyzing range queries on spatial data. In *ICDE'00*. Washington, DC, February 28 - March 3, 525–534.

LIN, X., LIU, Q., YUAN, Y., AND ZHOU, X. 2003. Multiscale histograms: Summarizing topological relations in large spatial datasets. In *VLDB'03*. Berlin, September 9-12, 814–825.

LIPTON, R. J., NAUGHTON, J. F., AND SCHNEIDER, D. A. 1990. Practical selectivity estimation through adaptive sampling. In *SIGMOD'90*. Atlantic City, NJ, May 23-25, 1–11.

LO, M.-L. AND RAVISHANKAR, C. V. 1996. Spatial hash-joins. In *SIGMOD'96*. Montréal, Québec, June 4-6, 247–258.

MAMOULIS, N. AND PAPADIAS, D. 1999. Integration of spatial join algorithms for processing multiple inputs. In *SIGMOD'99*. Philadelphia, Pennsylvania, June 1-3, 1–12.

MATIAS, Y., VITTER, J. S., AND WANG, M. 1998. Wavelet-based histograms for selectivity estimation. In *SIGMOD'98*. Seattle, Washington, June 2-4, 448–459.

MICALI, S. AND VAZIRANI, V. V. 1980. An $o(\sqrt{|V||E|})$ algorithm for finding maximum matching in general graphs. In *FOCS'80*. Syracuse, New York, October, 17–27.

PATEL, J. M. AND DEWITT, D. J. 1996. Partition based spatial-merge join. In *SIGMOD'96*. Montréal, Québec, June 4-6, 259–270.

POOSALA, V. 1997. Histogram-based estimation techniques in database systems. Ph.D. thesis.

SUN, C., AGRAWAL, D., AND ABBADI, A. E. 2002a. Exploring spatial datasets with histograms. In *ICDE'02*. San Jose, CA, February 26 - March 1, 93–102.

SUN, C., AGRAWAL, D., AND ABBADI, A. E. 2002b. Selectivity estimation for spatial joins with geometric selections. In *EDBT'02*. Prague, Czech Republic, March 25-27, 609–626.

SZALAY, A. S., KUNSZT, P. Z., THAKAR, A., GRAY, J., AND SLUTZ, D. R. 2000. Designing and mining multi-terabyte astronomy archives: The sloan digital sky survey. In *SIGMOD'00*. Dallas, Texas, May 16-18, 451–462.

TAO, Y., KOLLIOS, G., CONSIDINE, J., LI, F., AND PAPADIAS, D. 2004. Spatio-temporal aggregation using sketches. In *ICDE'04*. Boston, MA, March 30 - April 2, 214–226.

TAO, Y., SUN, J., AND PAPADIAS, D. 2003. Selectivity estimation for predictive spatio-temporal queries. In *ICDE'03*. Bangalore, India, March 5-8, 417–428.

TIGER. 2000. Tiger/line files. Technical report, U.S.Census Bureau, Washington, DC.

ZIPF, G. K. 1949. *Human Behaviour and the Principle of Least Effort: an Introduction to Human Ecology*. Addison-Wesley.