The VLDB Journal
DOI 10.1007/s00778-010-0205-7

REGULAR PAPER

Performance analysis of a dual-tree algorithm for computing spatial distance histograms

Shaoping Chen · Yi-Cheng Tu · Yuni Xia

Received: 29 September 2009 / Revised: 5 July 2010 / Accepted: 1 October 2010

Abstract Many scientific and engineering fields produce large volume of spatiotemporal data. The storage, retrieval, and analysis of such data impose great challenges to database systems design. Analysis of scientific spatiotemporal data often involves computing functions of all point-to-point interactions. One such analytics, the Spatial Distance Histogram (SDH), is of vital importance to scientific discovery. Recently, algorithms for efficient SDH processing in large-scale scientific databases have been proposed. These algorithms adopt a recursive tree-traversing strategy to process point-to-point distances in the visited tree nodes in batches, thus require less time when compared to the brute-force approach where all pairwise distances have to be computed. Despite the promising experimental results, the complexity of such algorithms has not been thoroughly studied. In this paper, we present an analysis of such algorithms based on a geometric modeling approach. The main technique is to transform the analysis of point counts into a problem of quantifying the area of regions where pairwise distances can be processed in batches by the algorithm. From the analysis, we conclude that the number of pairwise distances that are left to be processed decreases exponentially with more levels of the tree visited. This leads to the proof of a time complexity lower than the quadratic time needed for a brute-force algorithm and builds the foundation for a constant-time approximate algorithm. Our model is also general in that it works for a wide range of point spatial distributions, histogram types, and space-partitioning options in building the tree.

Keywords Scientific databases · Correlation function · Quad-tree · Spatial distance histogram

1 Introduction

The development of advanced experimental devices and computer simulations have given rise to explosive rendering of data in almost all scientific fields. As a result, scientific data management has gained much momentum in the database research community. Recent years have witnessed increasing interest in developing database systems for the management of scientific data [11,13,15,19,23,33,39,40]. While taking advantage of the optimized I/O and querying power of relational DBMSs, such systems still fall short of algorithms and strategies to satisfy the special needs of scientific applications, which are very different from those in traditional databases in their data types and query patterns. In this paper, we are interested in query processing against *scientific spatiotemporal data*. Such data are very popular in various scientific [2,14,31] and engineering [22] fields where natural systems (e.g., cells, galaxies) are often studied by computer simulations performed on the level of basic system components (e.g., atoms, stars). By nature, such

Work was done when Chen was a visiting professor at the University of South Florida.

S. Chen
Department of Mathematics, Wuhan University of Technology,
122 Luosi Road, 430070 Wuhan, Hubei,
People's Republic of China
e-mail: chensp@whut.edu.cn

Y.-C. Tu (✉)
Department of Computer Science and Engineering,
The University of South Florida, 4202 E. Fowler Ave.,
ENB118, Tampa, FL 33620, USA
e-mail: ytu@cse.usf.edu

Y. Xia
Computer and Information Science Department,
Indiana University-Purdue University Indianapolis,
723 W. Michigan St., SL280, Indianapolis, IN 46202, USA
e-mail: yxia@cs.iupui.edu

Journal: 778 Article No.: 0205 MS Code: VLDB-D-09-00108.2 ☐ TYPESET ☐ DISK ☐ LE ☐ CP Disp.:2010/10/20 Pages: 24 Layout: Large

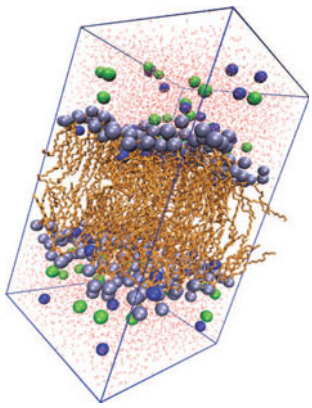


Fig. 1 A simulated hydrated dipalmitoylphosphatidylcholine bilayer system. We can see two layers of hydrophilic head groups (with higher atom density) connected to hydrophobic tails (lower atom density) are surrounded by water molecules (*red dots*) that are almost uniformly distributed in space

applications generate very large datasets. For example, molecular simulations often deal with systems with up to millions of atoms (see Fig. 1 for an example). In an extreme case, the Virgo consortium recently accomplished a simulation that consists of 10 billion stars [30].

Apart from the challenges of data storage/retrieval imposed by the gigantic volume of scientific data, we also face the issue of designing efficient algorithms for data querying and analysis. Scientific data analysis often requires computation of mathematical (statistical) functions [11,17] whose complexity goes beyond simple aggregates, which are the only analytics supported by modern DBMSs. Many complex analytics in scientific applications are found to be hierarchical in that they are often defined on top of a small number of low-level analytics as building blocks. Therefore, it is desirable to have built-in support for efficient processing of such low-level analytics in the DBMS. One salient example of such analytics is the *n-body correlation functions* (n-BCF). Generally, an n-BCF is a statistical measure of all the n-point subsets of the whole dataset. In a dataset with N data points, an n-BCF requires $O(N^n)$ time to compute in a brute-force way.

One type of 2-BCF query called the *Spatial Distance Histogram* (SDH) is of vital importance in computational sciences and thus the focus of this paper. The SDH problem can be formally stated as follows.

> Given the coordinates of N particles in a (2D or 3D) metric space, draw a histogram that represents the distribution of the pairwise distances between the N points.

The histogram has a single parameter l, which is the total number of buckets. Since the dataset is always generated from a system with fixed dimensions, the maximum distance between any two points $L_{\max}$ is also fixed. We often deal with *standard SDHs* whose buckets are of the same width. The width of the buckets (i.e., histogram resolution) $p = L_{\max}/l$ is often used as the parameter of the query instead. In other words, SDH asks for the counts of pairwise distances that fall into ranges $[0, p), [p, 2p), \ldots, [(l-1)p, lp]$, respectively. Basically, SDH is a discrete representation of a continuous 2-BCF called *Radial Distribution Functions* (RDF) [4,31]. The latter is required for the computation of many critical high-level analytics such as pressure, energy, [14] and structure factor [12]. Without RDF, meaningful analysis of the physical/chemical features of the studied natural system is not possible.

While a naive way to compute SDH takes $O(N^2)$ time, more efficient algorithms have been proposed in our previous work [38] and in the data mining community [16,25]. As summarized in Sect. 2.3, the main idea of this type of algorithms is to derive the histogram by studying the distances between two clusters of particles instead of those between two individual points. The clusters are represented by nodes in a space-partitioning tree structure. Although different implementations exist in [16,25] and [38], such an approach can be abstracted into a recursive tree-based algorithm described in Sect. 3. Since the recursion always happens between two disjoint subtrees, these algorithms are called *dual-tree algorithms* [16]. While experimental results support the efficiency of such algorithms, their complexity has not been thoroughly studied. In this paper, we present an analytical model to accomplish quantitative analysis of the performance of this algorithm. The main technique is to transform the analysis of particle counts into a problem of quantifying the area of interesting geometric regions. Our analysis not only leads to a rigorous proof of the algorithm's time complexity but also builds the foundation for approximate algorithms [16,38]. With time complexity that depends only on a controlled error bound, such algorithms are the only practical solutions to SDH computation in large datasets. Although we focus on a specific 2-body correlation function, the dual-tree algorithm can be easily extended to handle higher-order correlation functions [25]. Furthermore, the significance of this work is not limited to scientific databases: the dual-tree algorithm is also used to process a series of queries useful in data mining, such as batch k-nearest neighbor, outliner detection, kernel density estimation, and k-means [16].

Paper organization This paper is organized as follows: in Sect. 2, we summarize the contributions of the paper via comparison to related work; in Sect. 3, we sketch the dual-tree algorithm; we present our basic analytical model in Sect. 4 and two important extensions of the model in Sect. 5; we show an analysis of the time complexity of the dual-tree algorithm in Sect. 6; we report experimental results in Sect. 7 and conclude our paper in Sect. 8.

Journal: 778 Article No.: 0205 MS Code: VLDB-D-09-00108.2 ☐ TYPESET ☐ DISK ☐ LE ☐ CP Disp.:2010/10/20 Pages: 24 Layout: Large

2 Related work and our contributions

2.1 Scientific data management

The scientific community has been in a transition from developing *ad hoc* data processing systems based on flat files to utilizing modern database technologies for data management tasks. There are a large number of scientific databases built on top of existing relational DBMSs. Well-known examples include the following: the GenBank[1] database provides public access to about 80 million gene sequences; the Sloan Digital Sky Survey [33] enables public access to more than 100 attributes of 200 million objects in the sky; the QBISM project [3] delivers a prototype of querying and visualizing 3D medical images; and the Stanford Microarray Database[2] is a portal for storing and querying gene expression data.

However, scientific data are different from traditional data in that: (1) the volume of scientific data can be orders of magnitude larger; (2) data are often multidimensional and continuous; and (3) queries against scientific data are more complex. While the basic database system architecture can still be adapted, the above differences impose significant challenges to DBMS design. To meet such challenges, the database community has taken two different paths. The first one is to address domain-specific data management issues by modifying particular modules of existing relational DBMSs. There is a series of work dedicated to various aspects such as I/O scheduling [24], query processing [9,28] and data provenance management [10]. Another thrust is to build a general-purpose platform from scratch to support a wide range of scientific applications [6,21,32]. The work presented in this paper falls into the first category by emphasizing efficient processing of a special yet highly useful analytical query.

2.2 Computation of force/potential fields

The SDH/RDF problem is often confused with another group of problems—the computation of force/potential fields in scientific simulations. Specifically, the physical properties of a system component (represented as a point in space) is determined by the force applied to it by all other points in the system. Therefore, to compute the force applied to all points, $O\left(N^2\right)$ time is required. Since the force/potential can be expressed as an empirical integration formula, much efforts have been devoted to efficient force computation from a numerical analysis viewpoint. Most of the research in this field are derived from two lines of work: the Barnes–Hut [5] algorithm that requires $O\left(N \log N\right)$ time, and the fast multipole algorithm [18] with linear time complexity. These methods utilize unique features of the force (e.g., symmetry and fast degradation with distance) to bound the computational errors. However, they provide little insights into the SDH problem as the latter lacks such features.

Another method based on well-separated pair decomposition (WSPD) was proposed by Callahan and Kosaraju [7]. A WSPD is a series of pairs of subsets of the data points. Each pair of subsets P_i and P_j is well separated: the distance between the smallest balls (with radius r) covering the particles in P_i and P_j is at least sr where s is a system-level parameter. Following the algorithm in [7] that also utilizes a space-partitioning data structure called *fair-split tree*, a WSPD can be built in $O(N \log N)$ time and there are only $O(N)$ such pairs of subsets that cover all pairs of particles. As a result, the force fields can be computed in $O(N)$ time, given the WSPD. It may look intuitive that a WSPD can also be used to compute SDH: for each subset pair, their point-to-point distances fall into the range $[rs, rs+4r]$; by carefully choosing s and r, we can fit this range into relevant buckets of the histogram. However, the pitfall here is: s is a configurable parameter of the WSPD construction algorithm while r is not—it can be any value in each pair of subsets. If we enforce a specific value for r, the $O(N)$ performance guarantee is lost. Therefore, it does not provide a shortcut to efficient SDH processing to use the WSPD. In summary, the difficulty of the SDH problem is to put distances into buckets with clearly defined boundaries (Sect. 6.1) while the WSPD can only be manipulated to work with fuzzy ranges.

2.3 Algorithms for efficient SDH computation

Despite the importance of SDH, efficient SDH processing has not been intensively studied. Popular simulation data analysis softwares such as GROMACS [20] still follow the brute-force way to compute SDH. In [34] and [35], the SDH is processed by dividing the simulation space into bins and treating each bin as a single entity and run quadratic algorithms on these bins. Such an approximate solution, while reducing the computation time, can obviously yield uncontrollable errors. One approach to get the exact SDH is to issue a series of range queries (i.e., one for each bucket) for each data point, taking advantage of the kd-trees for range queries. This method, so called the **single-tree algorithm**, was extended to the **dual-tree algorithm** where the kd-trees are still used [16]. In our previous work [38], we utilized the Quad/Oct-tree to divide the simulation space into equally-sized cells and used it explicitly for SDH processing. The main idea behind the dual-tree algorithm is to process clusters of particles to take advantage of the non-zero width of the SDH bucket. The name "dual-tree" comes from the fact that it always works on a pair of such clusters (i.e., subtrees) while

[1] http://www.ncbi.nlm.nih.gov/Genbank.

[2] http://smd.stanford.edu/.

Journal: 778 Article No.: 0205 MS Code: VLDB-D-09-00108.2 TYPESET DISK LE CP Disp.:2010/10/20 Pages: 24 Layout: Large

the single-tree algorithm on one data point and one subtree. Note that the kd-tree is equivalent to a Quad-tree if we assume the particles are uniformly distributed in space. However, it turns out the use of Quad-tree is critical to achieve rigorous analysis. In addition to convincing experimental results, both work reported results of some asymptotical analysis. However, the results in [16] come with no technical details at all while our earlier paper [38] only sketched the main analytical results.

2.4 Contributions of this work

This paper significantly extends [38] by introducing the models behind the analytical results. In summary, this paper makes the following contributions.

1. We present details of an analytical model based on a geometric modeling approach. Such content is not found in any previous work;
2. The results in [16] and [38] are strictly based on the assumption that particles follow a uniform spatial distribution in space. This assumption is obviously unreasonable in real simulation environments. We relax this assumption in this paper;
3. We present an extended model for performance analysis in 3D space; and
4. We extend the analysis to arbitrary space-partitioning parameters (i.e., node degree) in building the spatial tree.
5. We also show the dual-tree algorithm has the same time complexity in processing SDHs with variable bucket width.

3 The dual-tree algorithm

In this section, we present the main idea of the dual-tree SDH algorithm (DT-SDH). DT-SDH is an abstraction of both methods presented in [16] and [38]. With the assumption of uniform particle distribution, the kd-tree in [16] is equivalent to a region Quad-tree, which is explicitly used in [38] and also the abstracted DT-SDH algorithm.

The algorithm first divides the simulated space into a grid, each cell of which records the number of data points in it. We call such a grid a *density map* and density maps with different cell sizes have to be maintained. We therefore organize all point coordinates into a point region Quad-tree [27] with each node representing a cell (square for 2D data and cube for 3D) in space. Point counts of each cell are cached in the corresponding tree node. Those with zero point count are removed from the tree. The height of the tree (denoted as H) is determined in a way such that the *average* number of points in all possible leaf nodes is no smaller than a predefined threshold β. To be specific, we have

$$H = \left\lceil \log_{2^d} \frac{N}{\beta} \right\rceil \quad (1)$$

where d is the number of dimensions and 2^d is essentially the maximal degree of tree nodes.

```
Procedure ResolveTwoTrees (A, B)
1  if A and B are resolvable
2      add n_a n_b to the corresponding bucket
3  elseif A and B are not leaf nodes
4          for each child a of A
5              for each child b of B
6                  ResolveTwoTrees (a, b)
7  else
8      compute all point-to-point distances between A and B
       and add each pair to the corresponding bucket
```

Fig. 2 Procedure ResolveTwoTrees—core of the DT-SDH algorithm

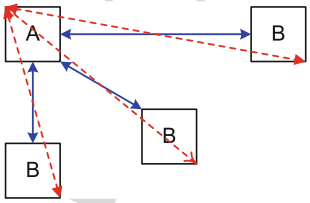


Fig. 3 Three scenarios in computing the minimum and maximum distance between two cells A and B, with solid (*dotted*) *line* representing minimum (maximum) distance in each case

The focal point of this algorithm is a procedure named ResolveTwoTrees (Fig. 2). To resolve two cells $\mathcal{A}$ and $\mathcal{B}$ (with total particle counts n_a and n_b, respectively), we first read the coordinates of the two cells and compute the range of distances between any pair of points, one from $\mathcal{A}$ and one from $\mathcal{B}$. Note that, given the coordinates of the two cells, this distance range can be computed in constant time (Fig. 3). If this range is contained in the range of a histogram bucket i, we say $\mathcal{A}$ and $\mathcal{B}$ are *resolvable* and they *resolve into* bucket i. In this case, we simply increment the count of bucket i by $n_a n_b$ (line 2). If the two cells are not resolvable, we recursively resolve all pairs of their child nodes (line 6). It is easy to see that, no matter how small the cells are in a density map, non-resolvable cell pairs always exist. Therefore, when we reach the lowest level of the tree, we have to calculate all distances of the particles in the unresolved cells (line 8).

In practice, β is set to be around 2^d. The intuition behind that is, when a pair of non-resolvable cells contains less than 16 (64 for 3D) distances (i.e., roughly 4 points in each cell), it does not help to further divide them. The process of tree construction can be accomplished in $O(N \log N)$ time.

Journal: 778 Article No.: 0205 MS Code: VLDB-D-09-00108.2 ☐TYPESET ☐DISK ☐LE ☐CP Disp.:2010/10/20 Pages: 24 Layout: Large

The algorithm starts from a certain level of the tree where the diagonal of the cells is no greater than the bucket width p. We denote this level as density map DM_0. In other words, we require

$$\delta \leq \frac{p}{\sqrt{d}} \tag{2}$$

where δ is the side length of the cells in DM_0 and d is the number of dimensions in the data. Note that no cells can be resolved if the above inequality does not hold true. The dual-tree algorithm runs as: first, all intra-cell particle-to-particle distances on DM_0 can be put into the first bucket $[0, p)$, as p is larger than the cells' diagonal length; second, ResolveTwoTrees is executed for all pairs of non-empty cells on DM_0.

3.1 Basic ideas in analyzing DT- SDH

The running time of DT- SDH is consumed by the following two types of operations:

(i) checking if two cells are resolvable (i.e., line 1 in ResolveTwoTrees); and
(ii) distance calculation for data points in cell pairs that are non-resolvable even on the finest density map (i.e., line 8 in ResolveTwoTrees).

When compared to the brute-force algorithm, we perform type (i) operations in hope of handling multiple distances in one shot such that the number of type (ii) operations is minimized. Given a histogram bucket width p, we start from a density map DM_0 with c cells. Thus, there are $O(c^2)$ type (i) operations to be performed on level DM_0. On the next map DM_1, there are $4 \times 4 = 16$ times of cell pairs to resolve. However, some of the cells in DM_1 do not need to be considered as their parents are resolved on DM_0. From this, we can easily see that the running time has something to do with p since it determines the number of cells in DM_0. However, in analyzing the time complexity of DT- SDH, we are interested in how the running time increases as the total number of points N increases, as p is a fixed query parameter. Qualitatively, as N increases, the height of the Quad-tree also increases (due to a fixed β), giving rise to a higher percentage of resolvable cell pairs on the leaf level. On the other hand, the total number of cell pairs also increases (quadratically). An essential question our analysis needs to answer is: given a cell **A** on DM_0, how many pairs of points are contained by those resolvable cells related to **A** as we visit more and more levels of density maps? Although this apparently has something to do with the spatial distribution of the points, our main strategy is to first analyze *how much area are covered by the resolvable cells* to simplify the process, and then discuss the effects of particle spatial distribution on this basic analysis. In the following section, we use a geometric modeling approach to quantify the area of resolvable cells of interest. Some of the symbols used throughout this paper are listed in Table 1.

Table 1 Notations and definitions

Symbol	Definition
N	Total number of particles in data
l	Total number of histogram buckets
p	Width of histogram buckets
m	An index of the density map (level on the Quad-tree)
i	An index on histogram buckets
δ	Side length of the cells on DM_0
$\alpha(m)$	Non-covering factor on level DM_m
S	The area of some region
s	Tiling factor
d	Number of dimensions in data (up to 3)

4 Main analytical results

4.1 Overview of our approach

Given any cell **A** on density map DM_0, our analysis first quantifies the area of a theoretical region containing all particles that can possibly resolve into the ith bucket with any particle in **A**. We call this region the *bucket i region* of cell **A** and denote it as $\mathbf{A_i}$. In a 2D example illustrated in Fig. 4, a cell **A** is drawn with four corner points O, O_1, O_2, and O_3, and $\mathbf{A_1}$ is bounded by curves and line segments connected by points C_1 through C_8. In our analysis, we consider the

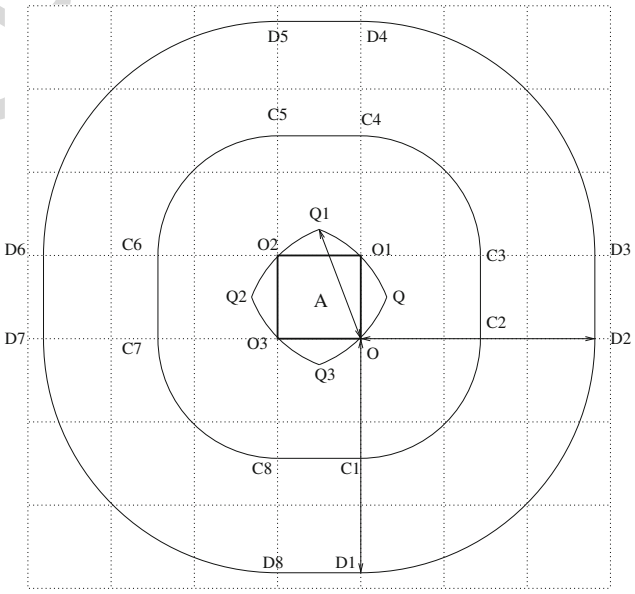


Fig. 4 Boundaries of bucket 1 and bucket 2 regions of cell **A**, with the bucket width p being exactly $\sqrt{2}\delta$. Here we show arcs $\widehat{Q_1Q_2}$, $\widehat{C_1C_2}$, and $\widehat{D_1D_2}$, all of which are centered at point O

Table 2 Values of $\alpha(m+1)/\alpha(m)$ in 2D space under different values of m and l

Map levels	Total number of histogram buckets (l)						
	2	4	8	16	32	64	256
$m=1$	0.508709	0.501837	0.50037	0.50007	0.500012	0.500002	0.5
$m=2$	0.503786	0.500685	0.500103	0.500009	0.499998	0.499999	0.5
$m=3$	0.501749	0.500282	0.500031	0.499998	0.499997	0.499999	0.5
$m=4$	0. 500838	0.500126	0.50001	0.499997	0.499998	0.499999	0.5
$m=5$	0. 50041	0.500059	0.500004	0.499998	0.499999	0.5	0.5
$m=6$	0.500203	0.500029	0.500002	0.499999	0.499999	0.5	0.5
$m=7$	0.500101	0.500014	0.500001	0.499999	0.5	0.5	0.5
$m=8$	0.50005	0.500007	0.5	0.5	0.5	0.5	0.5
$m=9$	0.500012	0.500003	0.5	0.5	0.5	0.5	0.5
$m=10$	0.500025	0.500002	0.5	0.5	0.5	0.5	0.5

Computed with Mathematica 6.0 based on the formulae generated in Sect. 4.4. Precision up to the 6th digit after decimal point

boundary situation of formula (2): the side length of cell **A** is set to be exactly $\delta = \frac{p}{\sqrt{2}}$. As we can easily see later, the case of $\delta < \frac{p}{\sqrt{2}}$ will not change the analytical results. Technical details on the quantification of the area of $\mathbf{A_i}$ is presented in Sect. 4.2.

The cells that are resolvable into bucket i with any subcells in **A** also form a region. We call such region the *coverable region* and denote it as $\mathbf{A'_i}$. Due to the shape of subcells, the boundary of such regions shows a zigzag pattern, as represented by solid blue lines in Fig. 6. When DT- SDH visits more levels of the tree, the resolution of the density map increases, and the boundary of region $\mathbf{A'_i}$ approaches that of $\mathbf{A_i}$. The quantification of the coverable regions' area is discussed in Sect. 4.3.

With the above results, we then study the area of coverable regions over all buckets and how the density map resolution affects it. Specifically, we define the ratio of $\sum_i \mathbf{A'_i}$ to $\sum_i \mathbf{A_i}$ as the *covering factor*. This is a critical quantity in our analysis as it tells how much area are "covered" by resolved cells. Obviously, the covering factor increases when we visit more levels of density map. Of special interest to our analysis is the *non-covering factor*, which represents the percentage of area that is not resolvable. The details about covering factor can be found in Sect. 4.4. A very important feature of the non-covering factor can be summarized in the following theorem.

$$g(i) = \begin{cases} (2\pi + 4\sqrt{2} + 1)\delta^2 & i = 1 \\ \left[2\pi i^2 + 4\sqrt{2}i \right. & \\ \quad -(i-1)^2\left(8\arctan\sqrt{8(i-1)^2-1} - 2\pi\right) & \\ \left. \quad + \sqrt{8(i-1)^2-1}\right]\delta^2 & i > 1 \end{cases} \tag{3}$$

Theorem 1 *Let DM_0 be the first density map where the* DT- SDH *algorithm starts running, and $\alpha(m)$ be the percentage of pairs of cells that are not resolvable on the density map that lies m levels below DM_0 (i.e., map DM_m). We have*

$$\lim_{p\to 0} \frac{\alpha(m+1)}{\alpha(m)} = \frac{1}{2}.$$

Proof The proof is developed in the remainder of this section starting from Sect. 4.2.

While shown in the form of a limit under large l (i.e., small p), Theorem 1 also works well under small l values. This can be effectively verified by numerical results obtained from the closed-form formulae we derive Eq. (9) and Eq. (10) to accomplish the proof. In Table 2, we can easily see that the ratio of $\alpha(m+1)$ to $\alpha(m)$ quickly converges even when l is very small.

Theorem 1 is important in that it shows the number of non-resolvable cell pairs decreases exponentially (by half) when more levels of the tree are visited. In RESOLVETWOTREES, if a cell pair is not resolved, we have to make 16 recursive calls to the same routine for the 4 children of each cell. Theorem 1 says that we can expect $16 \times 0.5 = 8$ pairs of the child nodes to be resolvable. For these resolved cell pairs, there is no need to further explore the pair of subtrees rooted by them. This greatly eases our analysis of the time complexity of DT-SDH (Sect. 6).[3] Now let us consider a formal proof of Theorem 1.

4.2 Maximal bucket region

As mentioned earlier, the bucket 1 region for cell **A** in Fig. 4 is connected by C_1 through C_8. Specifically, C_1C_2, C_3C_4, C_5C_6, and C_7C_8 are all 90-degree arcs centered at the four corners of cell **A** and their radii are of the same value p; C_2C_3, C_4C_5, C_6C_7, and C_8C_1 are line segments. It is easy to see that the area of this region is $\pi p^2 + 4p\delta + \delta^2$. Let us continue to consider distances that fall into the second bucket (i.e., [p, 2p]). Again, the bucket 2 region of **A** is of similar shape to the bucket 1 region except the radii of the arcs are $2p$, as drawn in Fig. 4 with a curve connected by points $D_1, D_2, \ldots, D_8$. However, points that are too close

[3] The techniques to derive Theorem 1 are important. However, readers can get a big picture of this work by browsing Theorem 2 (a more general form of Theorem 1) in Sect. 5.2 and then moving to Sect. 6.

Journal: 778 Article No.: 0205 MS Code: VLDB-D-09-00108.2 TYPESET DISK LE CP Disp.:2010/10/20 Pages: 24 Layout: Large

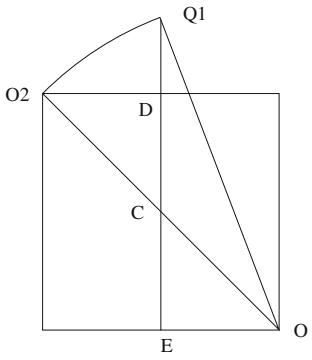


Fig. 5 A magnification of region **B** (i.e., $QQ_1Q_2Q_3$ formed by four arcs in Fig. 4). Here we only show arc Q_1O_2, which is a half of arc Q_1Q_2

to **A** can only resolve into bucket 1 since their distances to any point in **A** will always be smaller than p. These points are contained in a region as follows: on each corner point of **A**, we draw an arc with radius p on the opposite corner (i.e., arcs QQ_1, Q_1Q_2, Q_2Q_3, and Q_3Q). Therefore, the bucket 2 region should not include this inner region (denoted as region **B** hereafter, see Fig. 5 for a magnified illustration).

The area of the bucket 2 region is $\pi(2p)^2 + 8p\delta$ less the area of region **B**, which consists of eight identical smaller regions such as $\widehat{Q_1O_2}D$ (Fig. 5) and cell **A** itself. To get the area of $\widehat{Q_1O_2}D$, we first compute the magnitude of the angle $\angle Q_1OO_2$ as follows.

$$\angle Q_1OO_2 = \angle Q_1OE - \angle COE = \arctan\frac{Q_1E}{EO} - \frac{\pi}{4}$$

$$= \arctan\frac{\sqrt{p^2 - \left(\frac{\delta}{2}\right)^2}}{\frac{\delta}{2}} - \frac{\pi}{4}$$

Thus, the area of sector $\widehat{Q_1O_2}O$ is $\frac{1}{2}p^2\angle Q_1OO_2$. The area of region $\widehat{Q_1O_2}D$ can be obtained by the area of this sector less the area of triangles O_2DC and Q_1CO as follows:

$$S_{\widehat{Q_1O_2}D} = S_{\widehat{Q_1O_2}O} - S_{\triangle O_2DC} - S_{\triangle Q_1CO}$$

$$= \frac{1}{2}p^2\left[\arctan\frac{2\sqrt{p^2 - \left(\frac{\delta}{2}\right)^2}}{\delta} - \frac{\pi}{4}\right]$$

$$-\frac{\delta}{4}\sqrt{p^2 - \left(\frac{\delta}{2}\right)^2}$$

and we have $\pi(2p)^2 + 8p\delta - 8S_{\widehat{Q_1O_2}D} - S_{\mathbf{A}}$ as the area of the bucket 2 region.

The approach to obtain the area of bucket i ($i > 2$) regions is the same as that for bucket 2. For the area of the region formed by the outer boundary, we only need to consider that the arcs in Fig. 5 are of radii ip. Along with the fact $p = \sqrt{2}\delta$, our efforts lead to a general formula to quantify the area of the bucket i region in Eq. (3) shown on top of this page.

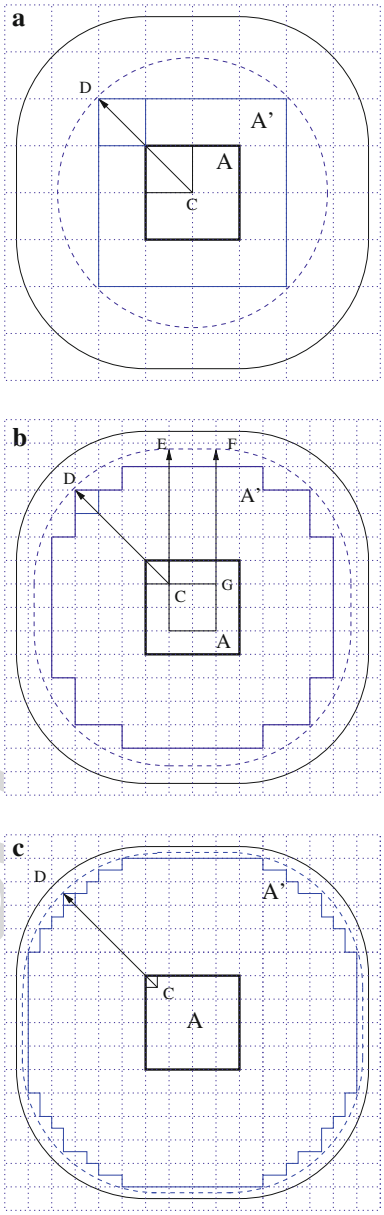


Fig. 6 Actual (*solid blue line*) and approximated (*dotted blue line*) coverable regions for bucket 1 under: a. $m = 1$; b. $m = 2$; and c. $m = 3$. Outer *solid black lines* represent the theoretical bucket 1 region. All *arrowed line* segments are drawn from the centers to the corresponding arcs with radius p

4.3 Coverable regions

The are two different scenarios to consider in deriving the area of coverable regions.

4.3.1 Case 1: the first bucket

Let us start our discussions on the situation of bucket 1. In Fig. 6, we show the coverable regions of three different density map levels: $m = 1, m = 2$, and $m = 3$, as

represented by blue-colored lines and denoted as $\mathbf{A}'$ in all subgraphs. Recall that, for two cells to be resolvable into bucket i, the minimum and maximum distance between them should both fall into range $[(i-1)p, ip]$. For $m = 1$, the resolvable cells are only those surrounding $\mathbf{A}$. All other cells, even those entirely contained by the bucket 1 region, do not resolve with any level 1 subcell of $\mathbf{A}$. As we increase m, the region $\mathbf{A}'$ grows in area, with its boundary approaching that of the bucket 1 region. To represent the area of $\mathbf{A}'$, the technique we adopt is to *develop a continuous line to approximate its boundary*. This technique will be used throughout our analysis. One critical observation here is: the furtherest cells in $\mathbf{A}'$ are those that can resolve with cells on the outer rim of $\mathbf{A}$. For example, the cell cornered at point D resolves with the cell cornered at point C in $\mathbf{A}$. If we draw a 90-degree arc centered at C, the arc goes through D and all cells on the northwestern corner of $\mathbf{A}'$ are bounded by this arc. To approximate the boundary of $\mathbf{A}'$, we can draw such an arc at all four corners of the graph and connect them with line segments (e.g., EF connecting the northwestern and northeastern arcs centered at point G in Fig. 6b), as shown by the blue dotted line. Obviously, this line approaches the theoretical boundary as m increases because the center of the arcs (e.g., point C) move further to the corner points of $\mathbf{A}$ as the cells become smaller. Note that this line gives rise to an optimistic approximation of $\mathbf{A}'$. In sect. 6, we will show that this overestimation will not harm our analysis on the time complexity of DT- SDH. The area of the coverable region for bucket 1 at level m can thus be expressed as

$$S_{\mathbf{A}'} = \pi p^2 + 4p\left(\delta - \frac{2\delta}{2^m}\right) + \left(\delta - \frac{2\delta}{2^m}\right)^2 \quad (4)$$

where the first item πp^2 is the area of the four 90-degree sectors centered at point C, the second item is the area of the four rectangles (e.g., EFGC in Fig. 6b) connecting the four sectors, and the last item is the area of the smaller square (e.g., the one with side CG in Fig. 6b) within cell $\mathbf{A}$.

4.3.2 Case 2: the second bucket and beyond

The cases of buckets beyond the first one are more complicated. First of all, the outer boundary of the bucket i ($i \geq 2$) regions can be approximated using the same techniques we introduced for bucket 1 (Sect. 4.3.1). To be specific, we can use the following generalized form of Eq. (4) to quantify the area of the region formed by the outer boundaries only.

$$S_{out}(i) = \pi(ip)^2 + 4ip\left(\delta - \frac{2\delta}{2^m}\right) + \left(\delta - \frac{2\delta}{2^m}\right)^2 \quad (5)$$

However, we also need to disregard the cells that lie in the inner boundary (e.g., those within or near region $\mathbf{B}$). This has to be considered in two distinct cases: $m = 1$ and $m > 1$.

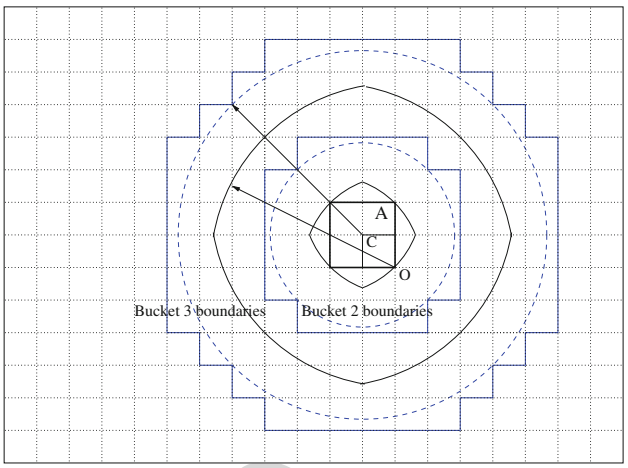


Fig. 7 Inner boundaries of the coverable regions of buckets 2 and 3 under $m = 1$. All *arrowed line* segments are of length $2p$

Let us first study the case of $m = 1$. Figure 7 shows examples with $m = 1$ with respect to the second and the third bucket. It is easy to see that any cell that contains a segment of the theoretical region $\mathbf{B}$ boundary will not resolve into bucket i because they can only resolve into bucket $i-1$. Furthermore, there are more cells that resolve into neither bucket $i-1$ nor bucket i. Here our task is to find a boundary to separate those $m = 1$ cells that can resolve into bucket i with any subcell in $\mathbf{A}$ and those that cannot. Such boundaries for buckets 2 and 3 are shown in Fig. 7 as solid blue lines. The boundary can be generated as follows: on each quadrant (e.g., northwest) of cell $\mathbf{A}$, we draw an arc (dotted blue line) centered at the corner point C of the furthest (e.g., southeast) subcell of $\mathbf{A}$ with radius $(i-1)p$. Any cell that contains a segment of this arc cannot resolve into bucket i (because they are too close to $\mathbf{A}$) but the cells beyond this line can. Therefore, we can also use these arcs to approximate the zigzagged real boundaries. Let us denote the region bounded by this approximate curve as region $\mathbf{B}'$. For $m = 1$, the arcs on all four quadrants share the same center C therefore they form a circle as region $\mathbf{B}'$. The radii of the circles are exactly $(i-1)p$ for bucket i. Note that this, again, could give rise to an optimistic approximation of the area of coverable regions. Therefore, the area of the coverable region for $m = 1$ and $i \geq 2$ is:

$$S_{\mathbf{A}'} = \pi(ip)^2 - \pi[(i-1)p]^2 \quad (6)$$

where the first item is the area of the region formed by the approximated outer boundary, which is given as a special case of Eq. (5) for $m = 1$ and happens to be a circle; and the second item is that of the region formed by the approximated inner boundary (i.e., region $\mathbf{B}'$).

For the case of $m > 1$, we can use the same technique described for the case of $m = 1$ to generate the curves to form region $\mathbf{B}'$. However, these curves are no longer a

Journal: 778 Article No.: 0205 MS Code: VLDB-D-09-00108.2 TYPESET DISK LE CP Disp.:2010/10/20 Pages: 24 Layout: Large

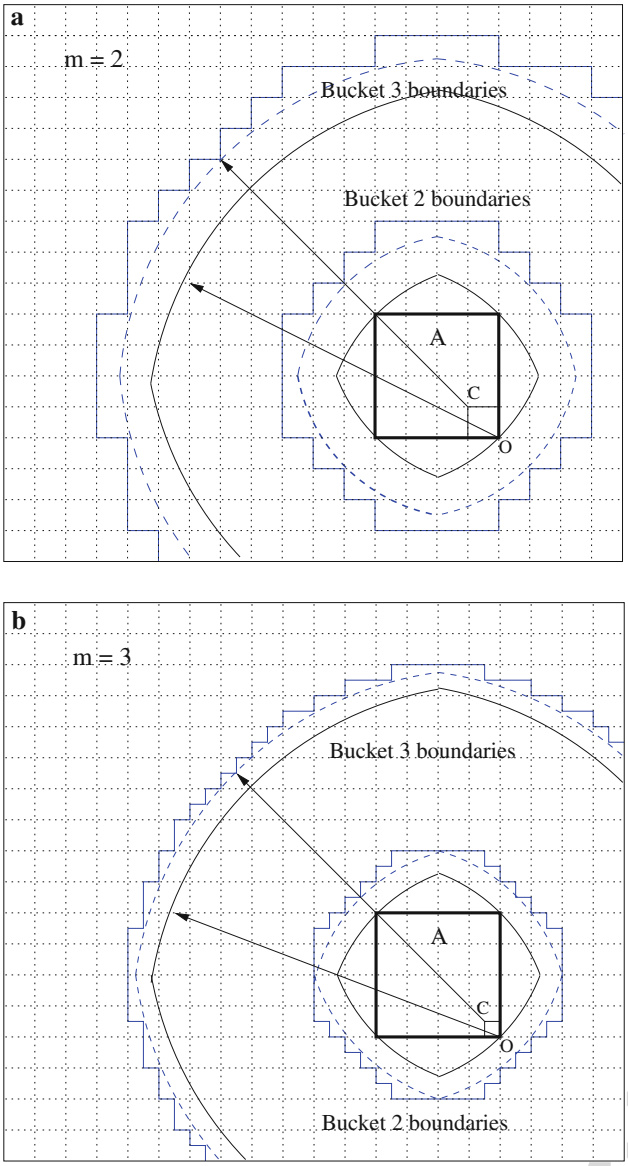


Fig. 8 Inner boundaries of the coverable regions of buckets 2 and 3 under $m = 2$ and $m = 3$. All arrowed line segments are of length $2p$

series of circles. In Fig. 8, we can find such curves for buckets 2 and 3 under m values of 2 and 3. As the four arcs on different quadrants no longer share the same center, the region $\mathbf{B}'$ boundaries (dotted blue lines) are of similar shapes to the theoretical region $\mathbf{B}$ boundaries (solid black lines). From the graphs, it is easy to see that the approximated curve fits the actual boundary better as m increases. Here we skip the formal proof as it is straightforward. Furthermore, it also converges to the region $\mathbf{B}$ boundary when m gets bigger. This is because the centers of the two arcs (with the same radii), points C and O, become closer and closer when the cell size decreases (as m increases).

The area of region $\mathbf{B}'$ (Fig. 9) can be computed in the same way as that of region $\mathbf{B}$. Following that, the area of coverable

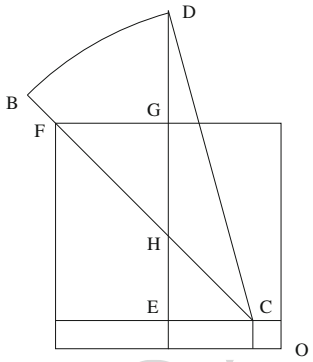


Fig. 9 An illustration on how to compute the area of region formed by four arcs in Fig. 8. Here we only show half of one of the arcs

region for $m > 1$ can be derived. The details of such results can be found in Appendix A. We define θ as a function of m for the convenience in further discussions:

$$\theta_m = \frac{1}{2} - \frac{1}{2^m}.$$

Let us denote the area of the coverable region $\mathbf{A}'$ for bucket i under different m values as $f(i, m)$. By combining and simplifying Eqs. (4), (6), and the results in Appendix A with $p = \sqrt{2}\delta$, we get Eq. (7) (as shown on top of this page), in which

$$\gamma_m = \sqrt{2(i-1)^2 - \theta_m^2}.$$

4.4 Covering factor and derivation of Theorem 1

In this section, we give a quantitative analysis on the relationship between $f(i, m)$ and the area of the theoretical region $g(i)$ for all buckets. For that purpose, given any density map level m, we define the *covering factor* $c(m)$ as the ratio of the total area of the coverable regions to that of the theoretical bucket i regions over all i. Relate this to Theorem 1, the more interesting quantity is the *non-covering factor*:

$$f(i,m) = \begin{cases} \left[2\pi + 4\sqrt{2} + 1 \right. & \\ \quad \left. - (8\sqrt{2}+4)\frac{1}{2^m} + \frac{4}{2^{2m}}\right]\delta^2 & i = 1, m \geq 1 \\ [2\pi(2i-1)]\,\delta^2 & i > 1, m = 1 \\ \left\{2\pi i^2 + 4\sqrt{2}i \right. & \\ \quad -(8\sqrt{2}i+4)\frac{1}{2^m} + \frac{4}{2^{2m}} & \\ \quad -8\left[(i-1)^2\left(\arctan\frac{\gamma_m}{\theta_m} - \frac{\pi}{4}\right)\right. & \\ \quad \left. \left. - \frac{1}{2}\theta_m(\gamma_m - \theta_m)\right] + 1\right\}\delta^2 & i > 1, m > 1 \end{cases} \quad (7)$$

$$\alpha(m) = 1 - c(m) = \frac{\sum_{i=1}^{l}[g(i) - f(i,m)]}{\sum_{i=1}^{l} g(i)} \quad (8)$$

With Eq. (8) and the results we have in Sects. 4.2 and 4.3, we are now ready to prove Theorem 1. Recall that we defined $\theta_m = \frac{1}{2} - \frac{1}{2^m}$ and $\theta_{m+1} = \frac{1}{2} - \frac{1}{2^{m+1}}$. Plugging Eqs. (3) and

Journal: 778 Article No.: 0205 MS Code: VLDB-D-09-00108.2 TYPESET DISK LE CP Disp.:2010/10/20 Pages: 24 Layout: Large

(7) into Eq. (8), we get $\frac{\alpha(m+1)}{\alpha(m)} = \frac{A(m)}{B(m)}$ where

$$A(m) = \frac{2}{2^m} - \frac{1}{4^m} + \frac{2^{\frac{3}{2}}}{2^m}(l + l^2) + \sum_{i=2}^{l} \sqrt{8(i-1)^2 - 1} - 4\sum_{i=2}^{l} \theta_{m+1} \sqrt{2(i-1)^2 - \theta_{m+1}^2} + 8\sum_{i=2}^{l} (i-1)^2 \arctan \frac{\sqrt{8(i-1)^2 - \theta_{m+1}^2}}{\theta_{m+1}} - 8\sum_{i=2}^{l} (i-1)^2 \arctan \sqrt{8(i-1)^2 - 1} \quad (9)$$

and

$$B(m) = \frac{4}{2^m} - \frac{4}{4^m} + \frac{2^{\frac{5}{2}}}{2^m}(l + l^2) + \sum_{i=2}^{l} \sqrt{8(i-1)^2 - 1} - 4\sum_{i=2}^{l} \theta_m \sqrt{2(i-1)^2 - \theta_m^2} + 8\sum_{i=2}^{l} (i-1)^2 \arctan \frac{\sqrt{8(i-1)^2 - \theta_m^2}}{\theta_m} - 8\sum_{i=2}^{l} (i-1)^2 \arctan \sqrt{8(i-1)^2 - 1} \quad (10)$$

The case of $p \to 0$ is equivalent to $l \to \infty$. Despite their formidable length and complexity, $A(m)$ and $B(m)$ are found to bear the following feature

$$\lim_{l \to \infty} \frac{A(m)}{B(m)} = \frac{1}{2} \quad (11)$$

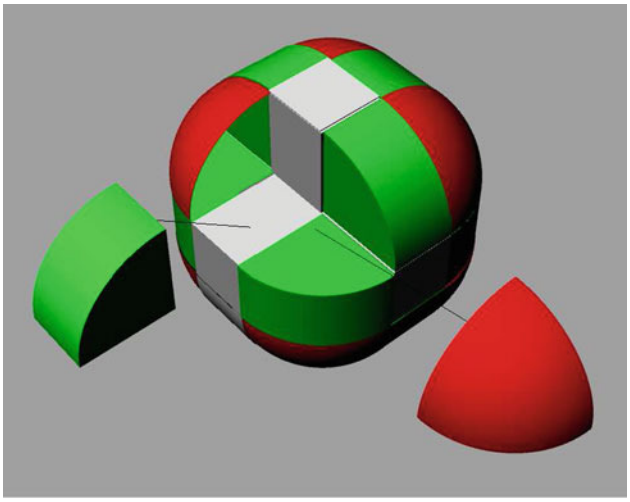
and this concludes the proof of Theorem 1. More details on derivation of Eq. (11) can be found in Appendix C.

5 Extensions

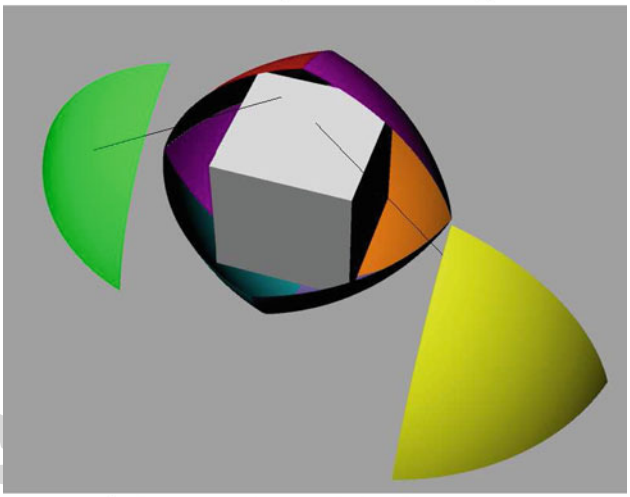
5.1 3D analysis

The strategies used to accomplish the analysis in Sect. 4 can be extended to 3D data. The outer and inner boundaries of bucket i regions are illustrated in Fig. 10. The analysis should be based on the volume of relevant regions surrounding a cube **A** with side length δ. The bucket 1 region (Fig. 10a) of **A** consists of the following components:

(1) quarter cylinders (green) with length δ and radius $p = \sqrt{3}\delta$;
(2) one-eighth of a sphere (red) with radius p;
(3) cuboids (white) with dimensions δ, δ, and p; and



a Outer boundary of the bucket 1 region.



b Inner boundary of the bucket 2 region.

Fig. 10 Geometric structures of the bucket 1 and bucket 2 regions for 3D data

(4) cube **A** itself, which is not shown in Fig. 10a, but can be seen in Fig. 10b.

There are eight pieces of each of the first two items and six pieces of item (3). The inner boundary (region **B**) of the bucket 2 region (Fig. 10b) consists of eight identical portions of a spherical surface centered at the opposite corner of **A** with radius p. Note that the projection of these regions on 2D are exactly those found in Fig. 4. Again, the shape of the region does not change with respect to bucket number i—we only need to change the radius from p to ip. The volume of the bucket i region can thus be expressed as

$$g(i) = \begin{cases} \frac{4}{3}\pi p^3 + 6p\delta^2 + 3\pi p^2\delta + \delta^3, & i = 1 \\ \frac{4}{3}\pi (ip)^3 + 6ip\delta^2 + 3\pi (ip)^2\delta + \delta^3 & \\ \quad -v(i, p, \delta), & i > 1 \end{cases}$$

Journal: 778 Article No.: 0205 MS Code: VLDB-D-09-00108.2 TYPESET DISK LE CP Disp.:2010/10/20 Pages: 24 Layout: Large

Table 3 Values of $\alpha(m+1)/\alpha(m)$ in 3D space under different values of m and l

Map levels	Total number of histogram buckets (l)						
	2	4	8	16	32	64	256
$m=1$	0.531078	0.509177	0.502381	0.500598	0.50015	0.500038	0.500002
$m=2$	0.514551	0.504128	0.50102	0.500247	0.50006	0.500013	0.5
$m=3$	0.505114	0.500774	0.500051	0.499987	0.499991	0.501551	0.500004
$m=4$	0.498119	0.497695	0.499076	0.499717	0.499931	0.498428	0.5
$m=5$	0.490039	0.49337	0.496703	0.499313	0.499811	0.499966	0.499983
$m=6$	0.47651	0.485541	0.49586	0.498521	0.499586	0.499897	0.499897
$m=7$	0.448987	0.469814	0.48972	0.497032	0.499241	0.499793	0.500138
$m=8$	0.38559	0.435172	0.478726	0.494029	0.49848	0.499448	0.5

Computed with Mathematica 6.0 based on formulae in Appendix B. Precision up to the 6th digit after decimal point

where the first four items in both cases represent the volume of the four components listed above and $v(i, p, \delta)$ is that for the region formed by half of a spherical surface in Fig. 10b. With $p = \sqrt{3}\delta$, the above equation becomes

$$g(i) = \begin{cases} \left(4\sqrt{3}\pi + 6\sqrt{3} + 9\pi + 1\right)\delta^3 & i = 1 \\ \left[4\sqrt{3}\pi i^3 + 6\sqrt{3}i + 9\pi i^2 + 1 - v(i,p)\right]\delta^3 & i > 1 \end{cases}$$

where $v(i, p) = 16V_{\mathbf{B}}$ and $V_{\mathbf{B}}$ is the volume of region **B** (see Appendix B for details).

We continue to develop formulae for the coverable regions $f(i, m)$ and non-covering factor $\alpha(m)$ as we do in Sects. 4.3 and 4.4. These formulae can be found in Appendix II of our technical report [37]. The complexity of such formulae hinders an analytical conclusion on the convergence of $\alpha(m+1)/\alpha(m)$ toward $\frac{1}{2}$. Fortunately, we are able to compute the numerical values of $\alpha(m+1)/\alpha(m)$ under a wide range of inputs. These results (listed in Table 3) clearly show that it indeed converges to $\frac{1}{2}$. This technique can be extended to higher dimensions and we conjecture that Theorem 1 still holds true. However, since the real simulation data has up to three dimensions, our analysis stops at 3D.

5.2 General tiling approach in space partitioning

In DT-SDH, the Quad-tree is built using a regular tiling [29] approach to partition the space, i.e., the subcells are of the same shape as the parent cell. In the previous analysis, for each node, we evenly cut each dimension by half, leading to 2^d partitions (child nodes) on the next level. However, in general, we could cut each dimension into $s > 2$ equal segments, giving rise to s^d equal-sized squares or cubes as in Fig. 11. In this section, we study how this affects the value of $\alpha(m)$.

First, the bucket i regions given by Eq. (3) are not affected. For the coverable regions, we incorporate the tiling factor s into the same reasoning as what we utilize to obtain Eq. (7). One exception here is the case of $m = 1, i \geq 2$:

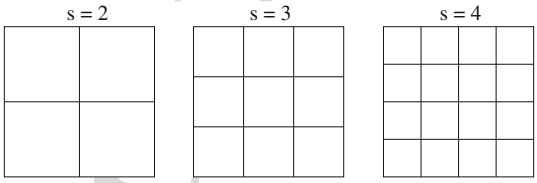


Fig. 11 Partitions of a 2D cell under different tiling factors

the approximate coverable region does not form a series of circles when $s > 2$, therefore Eq. (6) does not hold and this case should be handled in the same way as the case of $m > 1, i \geq 2$. Skipping the details, we get an improved version of Eq. (7) for $s > 2$ as Eq. (12), where $\theta'_m = \frac{1}{2} - \frac{1}{s^m}$ and $\gamma'_m = \sqrt{2(i-1)^2 - {\theta'_m}^2}$.

$$f(i,m,s) = \begin{cases} \left[2\pi + 4\sqrt{2} + 1 - (8\sqrt{2}+4)\frac{1}{s^m} + \frac{4}{s^{2m}}\right]\delta^2 & i = 1, m \geq 1 \\ \left\{2\pi i^2 + 4\sqrt{2}i - (8\sqrt{2}i+4)\frac{1}{s^m} + \frac{4}{s^{2m}} - 8\left[(i-1)^2\left(\arctan\frac{\gamma'_m}{\theta'_m} - \frac{\pi}{4}\right) - \frac{1}{2}\theta'_m\left(\gamma'_m - \theta'_m\right)\right] + 1\right\}\delta^2 & i > 1, m > 1 \end{cases} \quad (12)$$

With Eq. (12) to describe the coverable regions, we can easily generate new equations for the covering factor as a function of m and s. By studying these functions, we get the following theorem.

Theorem 2 *With a tiling factor s ($s \in \mathbb{Z}^+$), the non-covering factors have the following property*

$$\lim_{l\to\infty} \frac{\alpha(m+1)}{\alpha(m)} = \frac{1}{s}.$$

Proof The techniques to achieve this proof are very similar to those for Theorem 1. See Appendix D for the details.

Theorem 2 is obviously a nicely formatted extension of Theorem 1. Like Theorem 1, it is well supported by numerical results even under smaller values of l (details not shown in this paper). In Sect. 6, we will discuss the effects of s on the time complexity of DT-SDH.

6 Time complexity of DT-SDH

With Theorem 2, we achieve the following analysis of the time complexity of DT- SDH as a function of the input size N.

Theorem 3 *If the data points are uniformly distributed in space, the time complexity of* DT- SDH *under a general tiling factor s is $\Theta\left(N^{\frac{2d-1}{d}}\right)$ where $d \in \{2, 3\}$ is the number of dimensions of the data.*

Proof We derive the complexity of the algorithms by studying how the required time changes with the increase in system size N. Since the average number of particles in the leaf nodes is a constant β, one more level of tree will be built when N increases to $s^d N$. Therefore, we need to build a recurrence function that relates the running time under system size $s^d N$ to that under N.

We first study the time spent on operation (i) (i.e., resolving the cell pairs). We denote this time as T_c. For a given bucket width p, the starting level DM_0 is fixed in DT- SDH. Assume there are I pairs of cells to be resolved on DM_0, the total number of cell pairs becomes Is^{2d} on the next level DM_1. According to Theorem 2, only one s-th of the I pairs on DM_0 will not be resolved, leaving Is^{2d-1} pairs to resolve on DM_1. On level DM_2, this number becomes $Is^{2d-1}\frac{1}{s}s^{2d} = Is^{2(2d-1)}$, and so on. Therefore, $T_c(N)$ can be expressed as the summation of numbers of cell pairs to resolve in all levels of the tree starting from DM_0:

$$T_c(N) = I + Is^{2d-1} + Is^{2(2d-1)} + \cdots + Is^{n(2d-1)} = \frac{I\left[s^{(2d-1)(n+1)} - 1\right]}{s^{2d-1} - 1} \tag{13}$$

where n is the total number of levels in the tree visited by the algorithm. The value of n increases by 1 when N increases to $s^d N$. Therefore, by revisiting Eq. (13), we have the following recurrence:

$$T_c(s^d N) = \frac{I\left[s^{(2d-1)(n+2)} - 1\right]}{s^{2d-1} - 1} = s^{2d-1}T_c(N) - o(1) \tag{14}$$

Based on the master theorem [8], the above recurrence gives

$$T_c(N) = \Theta\left(N^{\log_{s^d} s^{2d-1}}\right) = \Theta\left(N^{\frac{2d-1}{d}}\right).$$

Note that the above conclusion about operation (i) has nothing to do with the data distribution.

Now let us investigate the time complexity for performing operation (ii), i.e., pairwise distance calculation. Interestingly, we have similar results as in Eq. (14).

As shown in the derivation of Eq. (14), there are $Is^{n(2d-1)}$ pairs of leaf nodes to resolve, among which $Is^{n(2d-1)}\frac{1}{s} = Is^{n(2d-1)-1}$ will be unresolved and the pairwise distances of the particles in them need to be computed one by one. When system size increases from N to $s^d N$, the number of unresolved leaf node pairs (denoted as L) becomes $Is^{(n+1)(2d-1)-1}$. Thus, we get the following recurrence:

$$L(s^d N) = s^{2d-1}L(N),$$

which is essentially the same as Eq. (14) and we easily get

$$L(N) = \Theta\left(N^{\frac{2d-1}{d}}\right) \tag{15}$$

Note that $L(N)$ is the number of non-resolvable cell pairs. Due to the assumption of uniformly distributed data, the number of point-to-point distances in these cells also follows Eq. (15). In Sect. 6.1, we will show that this claim still holds true when the assumption of uniform data distribution is relaxed.

Putting the above results about operations (i) and (ii) together, we conclude that the time complexity of DT-SDH is $\Theta\left(N^{\frac{2d-1}{d}}\right)$.

We have mentioned that our analysis is done based on an overestimation of the coverable regions on each density map, and the estimation error decreases as m increases. Relate this to Theorem 2, we have an underestimated non-covering factor α on each level. Since the estimation is more accurate on larger m, the real ratio of $\alpha(m+1)$ to $\alpha(m)$ can only be smaller than the one given by Theorem 2, making $\frac{1}{s}$ an upper bound. As a result, the complexity of the DT-SDH algorithm becomes $O\left(N^{\frac{2d-1}{d}}\right)$.

Note that the time complexity has nothing to do with the tiling factor s. In practice, we prefer smaller s values. Recall that the first map DM_0 should be the first level with cell size $\delta \leq p/\sqrt{d}$. With a bushy tree as a result of large s value, the cell size decreases more dramatically and we could end up a DM_0 with cell size way smaller than $p/\sqrt{d}$, giving rise to more cells to resolve (Eq. (13)).

6.1 Effects of spatial distribution of data points

To prove Theorem 3, we need to transform Eq. (15) into one that describes the number of distance calculations in the unresolved leaf nodes. This is obviously true for uniformly distributed data, in which the expected number of points in a cell is proportional to the cell size. However, in this subsection, we will show that Theorem 3 can be true even if we relax the assumption of uniformly distributed data points.

Let us consider any pair of non-resolvable cells $\mathcal{A}$ (with point count a) and $\mathcal{B}$ (with point count b) on the leaf level DM_k of the tree. Note that we cannot say $a = b$ (due to the non-uniform data distribution), and we expect to have $T_k = ab$ distances to compute between these two cells. When the system size increases from N to $s^d N$, we build another level of density map DM_{k+1}, in which $\mathcal{A}$ and $\mathcal{B}$ are both divided into s^d cells. Figure 12 shows an example for $s = 2$ and

Journal: 778 Article No.: 0205 MS Code: VLDB-D-09-00108.2 TYPESET DISK LE CP Disp.:2010/10/20 Pages: 24 Layout: Large

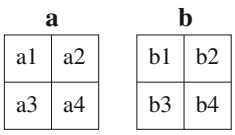


Fig. 12 Two non-resolvable leaf cells are divided into four subcells when data size increases by a factor of s^d

$d = 2$. Let us denote the original number of data points in the subcells as a_i ($i \in \{1, 2, \ldots, s^d\}$) and b_j ($j \in \{1, 2, \ldots, s^d\}$). In other words, we have $a = \sum_{i=1}^{s^d} a_i$ and $b = \sum_{j=1}^{s^d} b_j$. When N increases to $s^d N$, all values of a_i and b_j get a s^d-fold increase and the expected number of distance calculations becomes

$$T_{k+1} = \sum_{i,j} P_{i,j} s^d a_i s^d b_j \tag{16}$$

where $P_{i,j}$ is a binary variable that tells whether subcells i and j are non-resolvable on DM_{k+1}. Without any assumptions, we only know that the average of $P_{i,j}$ over all combinations of i and j is $\frac{1}{s}$ (Theorem 2). For Theorem 3 to be true, we need to show that

$$T_{k+1} \leq \frac{s^{2d}}{s} T_k = s^{2d-1} ab \tag{17}$$

6.1.1 Effects of a common point distribution

We first show that, if the distribution of data points is *cell-wise uniform* on density map DM_k, the condition specified in formula (17) is satisfied. Being cell-wise uniform means that the data are uniformly distributed within each cell, i.e., we have

$$a_1 = a_2 = \cdots = a_{s^d} = \frac{a}{s^d}$$

and

$$b_1 = b_2 = \cdots = b_{s^d} = \frac{b}{s^d},$$

which easily leads to

$$T_{k+1} = \frac{1}{s} \sum P_{i,j} s^d a s^d b = s^{2d-1} ab.$$

Being a less constrained assumption than system-wise uniform distribution (which also requires $a = b$), the cell-wise uniform distribution is a safe assumption in many scientific domains. This is because components of natural systems are generally not compressed arbitrarily to form high-density clusters due to the existence of chemical bonds or inter-particle forces [1,26]. As a result, data points tend to spread out "evenly", at least in a localized area. The water molecules is a good example of this. Note that we only need to make the assumption of cell-wise uniformity in the leaf nodes to make Theorem 3 true. In fact, we often found uniform regions on high-level tree nodes. For example, by studying the dataset illustrated in Fig. 1, we found that atoms are uniformly distributed in 61 out of 64 of the nodes on level 3 of the Quad-tree. Cell-wise uniformity is also a popular observation in many traditional spatiotemporal database applications [36].

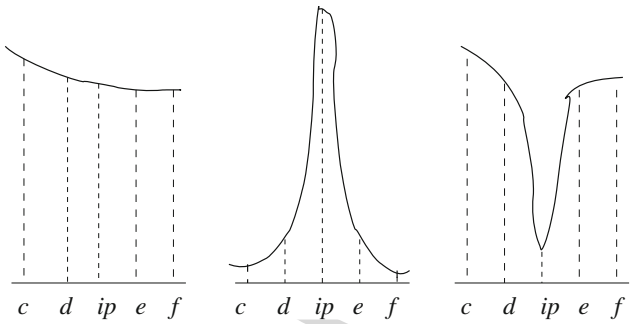


Fig. 13 Three cases of distribution of distances around the edge of buckets i and $i + 1$, with the *solid curves* representing portions of the density function of the distances; $[c, d]$ and $[e, f]$ are examples of distance ranges of resolvable subcells. Those of the non-resolvable subcells are not shown. *Line* segments are not drawn on scale. For example, ip does not have to be the middle point of $[c, f]$ in practice

6.1.2 More general conditions

A more general discussion on the necessary conditions of Theorem 3 would be helpful in identifying the limitations of DT- SDH. We believe that *skewed point distributions will affect the correctness of Theorem 3 only in rare cases*. Intuitively, a skewed point distribution can give rise to a skewed distance distribution. Revisiting Fig. 12 and Eq. (16), we can easily see that T_{k+1} is basically a sum of $s^{2d} a_i b_j$ weighted by the binary variable $P_{i,j}$, which has an average of $\frac{1}{s}$ according to Theorem 2. Therefore, the condition for Theorem 3 to hold true is that *there is no positive correlation between the occurrence of* $P_{i,j} = 0$ *and large values of* $a_i b_j$. In other words, as long as the peaks in the data distribution do not always coexist with the non-resolvable cell pairs, Theorem 3 will not be harmed. We know $P_{i,j}$ is determined solely by the geometry of the cells and p. If we model the data placement as a regular stochastic process (e.g., Zipf, mixed-Gaussian, ...) , the lack of correlation between $P_{i,j}$ and data distribution (on which the values of a_i, b_j depend) can be easily justified. An adversary can certainly generate cases to beat DT- SDH by adding more constraints to the data distribution. We will discuss that in Sect. 6.1.3.

Another way to describe the above condition is, as shown in the middle graph of Fig. 13, we cannot have high density of distances centering around (most or all of the) the bucket boundaries. Suppose two cells (e.g., $\mathcal{A}$ and $\mathcal{B}$ in Fig. 12) have a distance range $[c, f]$, which overlaps with buckets i and $i + 1$. With one more level of density map built, their subcell pairs could generate resolvable distance ranges such as $[c, d]$ and $[e, f]$ (because they do not contain ip—the boundary of

Journal: 778 Article No.: 0205 MS Code: VLDB-D-09-00108.2 TYPESET DISK LE CP Disp.:2010/10/20 Pages: 24 Layout: Large

the two buckets). It also generates non-resolvable distance ranges that contain ip. If the distribution of distances has high density around ip, most of the area under the density curve will fall into the non-resolvable ranges. On the contrary, if the density curve around ip is not a sharp peak (e.g., left graph in Fig. 13), we could have roughly equal amount of area under the resolvable and non-resolvable ranges. Or, in another extreme case (e.g., right graph of Fig. 13) where the density is very low around ip, most of the distances will be in the resolvable ranges.

Interestingly, there are easy remedies to the situation shown in the middle graph of Fig. 12. We can

(1) compute another SDH by moving the boundaries of all buckets to the left (or right) by $\frac{p}{2}$, or
(2) decrease the bucket width to γp where $0 < \gamma < 1.0$ and $\frac{1}{\gamma}$ is not an integer.

By both methods, we can generate a histogram that shows all the trends in the distance distribution (exactly what we need in a SDH) yet most of the distance calculations are avoided. In the second case, the SDH generated is of an even higher resolution. The technical details of designing such algorithms are beyond the scope of this paper.

6.1.3 *Counterexamples to Theorem 3*

In this subsection, we discuss data distribution patterns that serve as adversaries against DT- SDH. We have mentioned that Theorem 3 still holds true under cell-wise uniform distribution. Therefore, an adversary case would obviously involve tightly clustered data points. However, such clusters by themselves do not necessarily increase the time complexity of DT- SDH. To do that, additional conditions have to be satisfied.

Figure 14 illustrates a high-density cluster A in 2D space. To study the impact of cluster A on Theorem 3, we have to consider the locations of data points out of A. If the other data points spread out in the whole space, Theorem 3 would still be true as this is roughly the scenario of cell-wise uniform distribution. Therefore, it requires a large number of particles to be located in the non-coverable regions of A to make a "bad" case for DT- SDH. There can be two scenarios: 1) the other data points spread out in the non-coverable regions of A and 2) there are high-density clusters (e.g., B in Fig. 14) within the non-coverable regions. One thing to point out is: for the above scenarios to be effective adversaries, the points out of A must reside in a very narrow band. This is because the non-coverable regions shrink as the cells on the leaf nodes of the Quad-tree get smaller (due to the increase of N, as shown by Theorem 1).

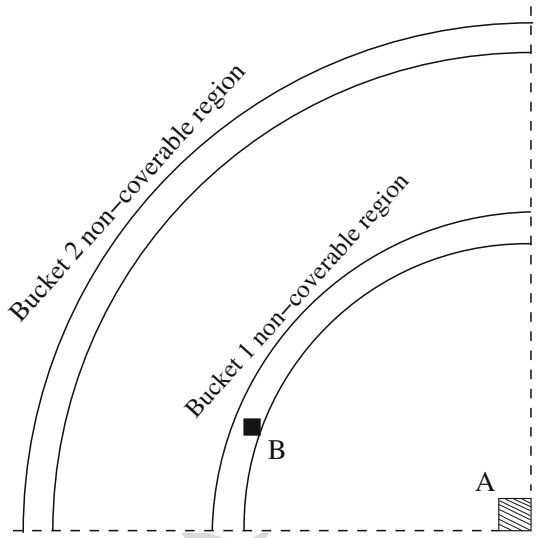


Fig. 14 A cell (*denoted* as cell A) with a large number of data points in 2D space and its first two non-coverable regions on the lowest level of the tree. The non-coverable regions are represented by annuli (*rings*). Only one quarter of the regions are plotted

From the above discussions, we have shown the following two conditions about data distribution are required for constructing an adversary input for DT- SDH.

(1) high-density data clusters must exist;
(2) at least one pair of such clusters are in each others' non-coverable regions.

Some examples of such datasets are shown in Fig. 15, in which a large number of particles are in high-density clusters, and the distances between pairs of clusters equal to ip where i is a positive integer. In an extreme case (top graph in Fig. 15) where the distance between any pair of clusters is ip, the particles are organized in a linear pattern. Fortunately, real scientific data will not likely follow such data distributions because the particles in nature tend to spread out in space (instead of forming clumps with a particular distance from each other). Again, all cases mentioned here can be easily handled by the remedies introduced in Sect. 6.1.2.

6.2 SDH with variable bucket width

So far, we have studied the performance of DT- SDH in computing a standard SDH in which all buckets are of the same width p. In this subsection, we extend our analysis to the processing of SDHs with variable bucket width. We denote p_i as the ending point of bucket i, i.e., bucket i covers the range $[p_{i-1}, p_i)$. Due to the variable bucket width, the results in Sect. 4 cannot be directly adopted to accomplish the analysis. Instead, we consider a variation of the DT- SDH algorithm (which we call DT- SDH') to compute the non-standard SDH

Journal: 778 Article No.: 0205 MS Code: VLDB-D-09-00108.2 TYPESET DISK LE CP Disp.:2010/10/20 Pages: 24 Layout: Large

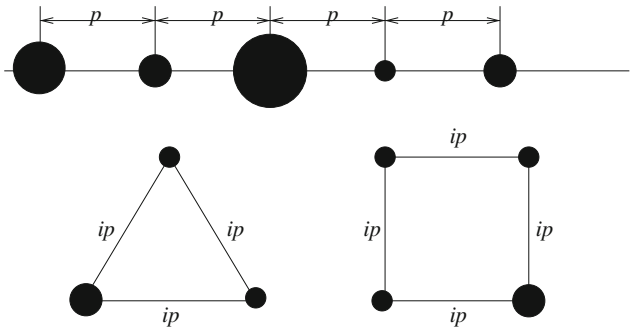


Fig. 15 Several patterns of particle spatial distribution that lead to large number of non-resolvable distances near the bucket boundaries. Each ball represents a cluster of particles. *Line* segments are not drawn to scale

and derive its time complexity. We then prove that the running time of DT- SDH is equivalent to that of DT- SDH'.

For a SDH with l buckets of variable size, DT- SDH' computes it in $l-1$ steps. In the ith step, we run DT- SDH to compute a SDH with only two buckets that are separated by p_i (i.e., the two buckets are $[0, p_i)$ and $[p_i, L_{max}]$). It is easy to see that the SDH of interest can be obtained from all such two-bucket SDHs. The only thing to point out is that, in each step of running DT- SDH, we choose the DM_0 based on the width of the smaller bucket (recall Eq. (2)).

Theorem 4 *The time complexity of running* DT- SDH' *for computing a non-standard SDH that divides the distance domain into two buckets is* $O\left(N^{\frac{2d-1}{d}}l\right)$ *where* $d \in \{2, 3\}$.

Proof The DT- SDH' runs DT- SDH for a total of $l-1$ times, each time it computes a two-bucket SDH. To prove the theorem, it is sufficient to show that the time complexity of DT- SDH on computing any two-bucket SDH is $O\left(N^{\frac{2d-1}{d}}\right)$— same as that for DT- SDH to computer a standard SDH. Without loss of generality, we denote the smaller one of the two bucket width as q and that of the other bucket as $r = L_{max} - q$. We can still use the techniques shown in Sect. 4 to analyze this, except we only need to consider two buckets $[0, q)$ and $[q, L_{max}]$. For the two buckets, we can then generate the area of the bucket regions $g(1)$ and $g(2)$, and that for the coverable regions $f(1, m)$ and $f(2, m)$. The formulae for the above area can be found in Appendix E. We then get the non-covering factor as

$$\alpha(m) = \frac{g(1) + g(2) - f(1, m) - f(2, m)}{g(1) + g(2)}$$

And the covering factor has the following feature

$$\frac{\alpha(m+1)}{\alpha(m)} \leq \frac{1}{2}.$$

The above is similar to Theorem 1 and we easily conclude the proof by following the path we took to prove Theorem 3.

The following theorem gives the time complexity of DT- SDH on computing a non-standard SDH.

Theorem 5 *The time complexity of running* DT- SDH *for computing a SDH with variable bucket width is also* $O\left(N^{\frac{2d-1}{d}}l\right)$ *where* $d \in \{2, 3\}$.

Proof We achieve the proof by comparing the number of operations in the DT- SDH to that in DT- SDH'. Specifically, we have the following observations:

(1) type (ii) operations: if a pair of points fall into a pair of non-resolvable leaf cells in DT- SDH, they are also in the same non-resolvable leaf cells in DT- SDH';
(2) type (i) operations: for any pair of cells, if they are visited by DT- SDH' for an attempt to resolve them, they are also visited by DT- SDH for the same purpose.

The above two facts show that the time spent by DT- SDH is no more than that by DT- SDH' to process the same dataset, and this concludes the proof.

7 Empirical evaluations

7.1 Experimental setup

We have implemented the DT- SDH algorithms using the C programming language and tested it with synthetic and real datasets. The experiments were run in an Apple Mac Pro workstation with two dual-core Intel Xeon 2.66 GHz CPUs, and 12 GB of physical memory. The operating system was OS X 10.5 Leopard.

The datasets used in our experiments include three groups of synthetic ones and data from real simulations. Among the synthetic data groups, one was generated following a uniform distribution of data points, one following a Zipf distributions with various orders, and one from mixed-Gaussian distributions. All point coordinates in the synthetic datasets are rendered in a 3D cube whose side length is 25,000 units. For the Zipf-based datasets, we divided the entire data space into a large number of small blocks (i.e., cubes with side length 5 to 50), and each small cube was assigned a random rank. Given two cubes with ranks i and j, the expected number of data points are of ratio $j^\alpha : i^\alpha$ where $\alpha \geq 1.0$ is the *order* of the Zipf distribution. The well-known fact is that, even with order 1.0, a Zipf distribution brings high level of skewness in the data. We also generated test data using the mixed-Gaussian model. Specifically, the data points are rendered from 3 to 5 normal distributions with a fixed standard deviation and means randomly chosen within a 2D simulation space. Each normal distribution carries the same weight, and we assume there is no correlation among the two dimensions.

Journal: 778 Article No.: 0205 MS Code: VLDB-D-09-00108.2 TYPESET DISK LE CP Disp.:2010/10/20 Pages: 24 Layout: Large

We used a series of C libraries provided by the randlib [4] package to generate relevant random numbers.

Another dataset was generated from a real molecular dynamics study to simulate a bilayer membrane lipid system in NaCl and KCl solutions, as illustrated in Fig. 1. The original dataset records the coordinates of 286,000 atoms over 10,000 time instances (data in each time step is called a *frame*). In order to make the experiments comparable to those using synthetic data, we randomly choose and duplicate atoms in this dataset to reach different dataset sizes N. Specifically, we combine the data from consecutive frames to create datasets with size greater than 286,000.

All experiments were run under a series of N values ranging from 100,000 to 25,600,000. In the following text, we report the results of three lines of experiments.

7.2 Model verification

The objective of this set of experiments is to evaluate the correctness of our basic analytical model, which gives Theorem 1 as the foundation of complexity analysis. Instead of verifying the values of $\alpha(m+1)/\alpha(m)$ listed in Tables 2 and 3, we focus on the number of distances in the non-resolvable cells under different m values to study how data distribution (especially the skewed ones) affects the effectiveness of our model. Ideally, the number of unresolved distances should follow the same pattern as described in Theorem 1—it should decrease by half every time m increases by 1.

Figure 16 shows the absolute number of resolved distances (plotted on a logarithmic scale) achieved. Each line represents one experiment on a dataset of a particular size. For all experiments, we can see that the line starts from a small value and then reaches the highest value on the following level. The first value in each line reflects those distances resolved on DM_0—it is small because it only contains those intra-cell distances that resolve into bucket 1. Starting from DM_1, the values drop at a rate that is close to $\frac{1}{2}$—this trend can be easily seen by comparing the slopes of the data lines to that of a standard function $y = c\left(\frac{1}{2}\right)^x$. One thing to point out is: the slopes of some of the data lines in Fig. 16 (e.g., the top lines in all 3D experiments) are even slightly smaller than that of the standard curve. This indicates that the distances are consumed in a higher rate than what we expect from our model. To better interpret the experimental results, we need to see from an opposite angle by showing how many distances are left unresolved on each level.

For the same experiments, Fig. 17 plots the percentage of unresolved distances on a logarithmic scale. Again, each line starts by the reading of DM_0 and we draw a standard line with slope $-\frac{1}{2}$ to indicate the expected trend given by Theorem 1. It is easy to see that the distances are resolved at a rate close to $\frac{1}{2}$. The only exceptions appear in the real 3D simulation data experiment (Fig. 17f) where the number of distances decrease at a slightly slower rate on the middle levels. But the final values all ended up below the standard line. Clearly, this is in conformity with Theorem 1, which says half of the uncovered area will be covered by going one level down the tree. In fact, a majority of the plotted values are below the corresponding standard lines, supporting our claim that Theorem 1 is actually a lower bound of the expected performance. The important information here is that the number of resolved distances shows the same trend for all datasets, indicating the robustness of our model. The skewed Zipf point distribution does not at all cause degraded performance. In fact, we found that, among the three datasets, it always took the least amount of time for DT-SDH to process the Zipf dataset. Here we hold the discussions on the effects of data skewness on running time till Sect. 7.4 where the results of more skewed datasets (Zipf, mixed-Gaussian) will be reported.

7.3 Efficiency of DT- SDH

The main purpose of this experiment is to verify the time complexity of DT- SDH. In Fig. 18, the running time of our algorithm is plotted against the size of 2D experimental datasets. Fig. 18a shows the results of uniformly distributed data and Fig. 18b for those following the Zipf distribution, and Fig. 18c for the real simulation data. Both the running time and data size are plotted on logarithmic scales; therefore, the slopes of the lines reflect the time complexity of the algorithms. For comparisons, we draw an identical dotted line in each graph with a slope of 1.5. Each point in the graphs shows the result of one single run of DT- SDH as the long running time under large N prohibits having multiple runs. However, we did run multiple experiments with different random seeds for the cases of smaller N and observed very little variances in running time.

The brute-force approach ('Dist') always shows an exact quadratic running time (i.e., the slope of the line is 2). The other lines (with spots) represent experiments using our algorithm under different bucket numbers l. Clearly, the running time of our algorithm grows less dramatically—they all have a slope of about 1.5. When bucket size decreases, it takes more time to run our algorithm, although the time complexity is still $\Theta(N^{1.5})$. The cases of large bucket numbers ('$l = 256$') are worth some attention: the running time is similar to that of the brute-force approach when N is small. As N increases, the slope of the line changes to around 1.5. The reason for this is: when N is small, we have a tree with very few levels; when the query comes with a very small bucket size p, we end up starting DT- SDH from the leaf level of the tree and have to essentially calculate most or all distances. However, the same query will get the chance to

[4] http://randlib.sourceforge.net/.

Journal: 778 Article No.: 0205 MS Code: VLDB-D-09-00108.2 TYPESET DISK LE CP Disp.:2010/10/20 Pages: 24 Layout: Large

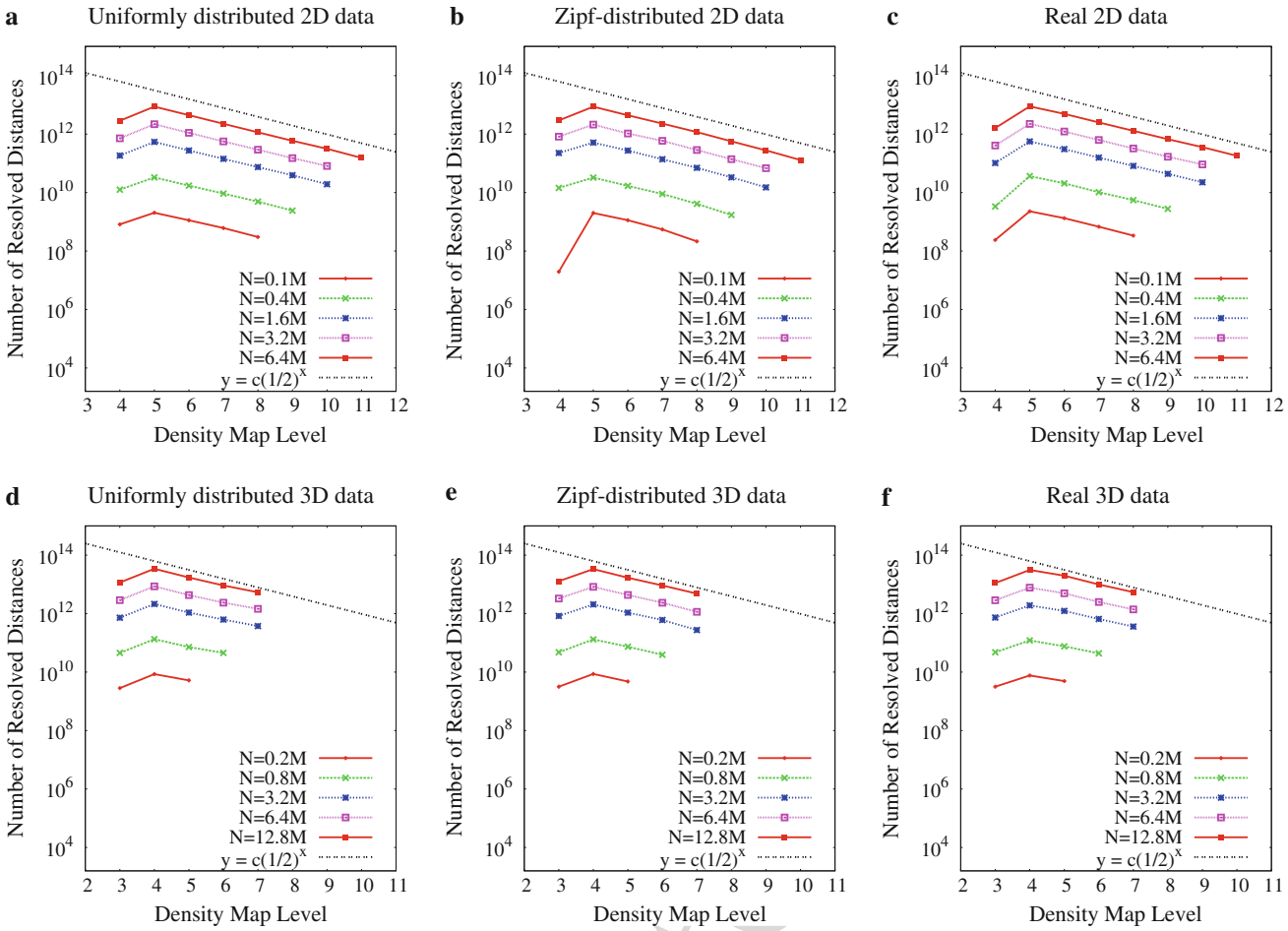


Fig. 16 Number of distances resolved on different levels of the tree

resolve more cells when the tree becomes taller, as a result of larger N. Again, the actual running time for the skewed dataset is always shorter than that for the uniform dataset with the same size. This can be seen by the relative positions of colored lines to the '$T = O(N^{1.5})$' line. The results of the real dataset are almost the same as those for the uniform data.

We have similar results for 3D data (Fig. 19): the corresponding lines for DT- SDH have slopes that are very close to $\frac{5}{3}$, confirming our asymptotic analysis. Again, the cases for large l values are worth more discussions. For '$l = 64$', the running time grows quadratically till N becomes fairly large (1,600,000) and then the slope of the line changes to $\frac{5}{3}$. One thing to notice is that the slope of the last segment of '$l = 64$' in Fig. 19b is almost 2. This does not mean the time complexity is going back to quadratic. In fact, it has something to do with the zigzag pattern of running time change in the Zipf data: for three consecutive doubling N values (i.e., a 8-fold increase), the running time increases by 2, 4, and 4 times, which still gives a $2 \times 4 \times 4 = 32$ fold increase in total running time (instead of a 64-fold increase in a quadratic algorithm).

7.4 Effects of skewed data distribution

To further test the effects of skewed datasets on the performance of DT- SDH, we run 2D experiments using data generated from Zipf distribution of different orders and the mixed-Gaussian distributions with different standard deviations (SD). By increasing the order of Zipf or decreasing the SD of the mixed-Gaussian, we are supposed to generate more skewed datasets as more points will be concentrated on smaller regions. In these experiments, we computed a histogram with bucket width 4419.0.[5] Four random seeds were used to generate data of different sizes ranging from 100,000 to 25,600,000. Thus, for a particular N (under one Zipf order), we tested the algorithm with four datasets.

The results of the Zipf datasets are shown in Fig. 20, in which both data size and running time are plotted on logarithmic scales. The same experiments were run under two block sizes (50 and 5), representing two levels of "tightness" of the

[5] This is exactly the diagonal of cells on the 4th level of the tree. We chose a relatively large p to save the total experimental time. We believe it is sufficient to show the trends.

Journal: 778 Article No.: 0205 MS Code: VLDB-D-09-00108.2 TYPESET DISK LE CP Disp.:2010/10/20 Pages: 24 Layout: Large

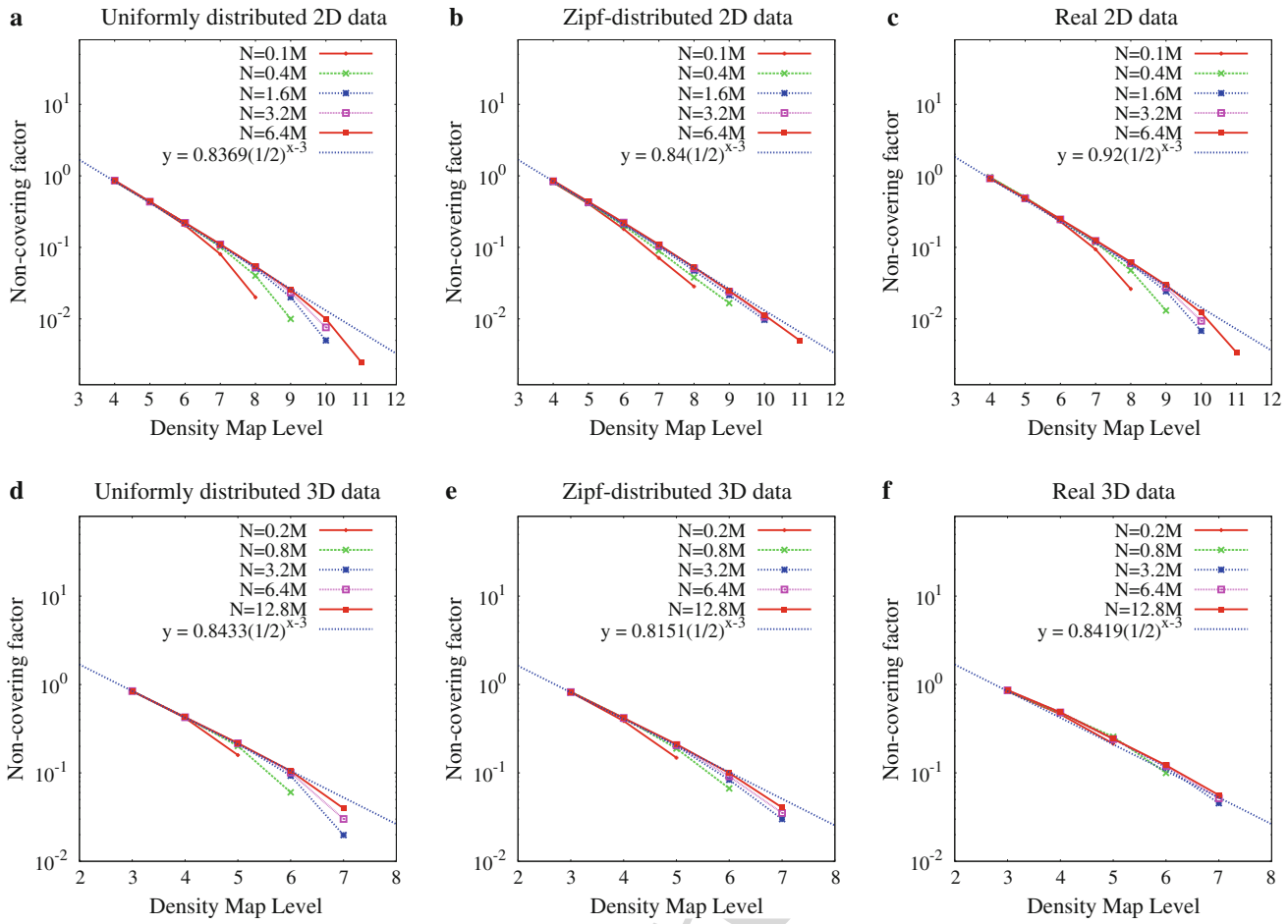


Fig. 17 Non-covering factors upon visiting different levels of the tree. Here the factor is calculated as the ratio of number of unresolved distances to total number of distances after visiting m levels in the tree

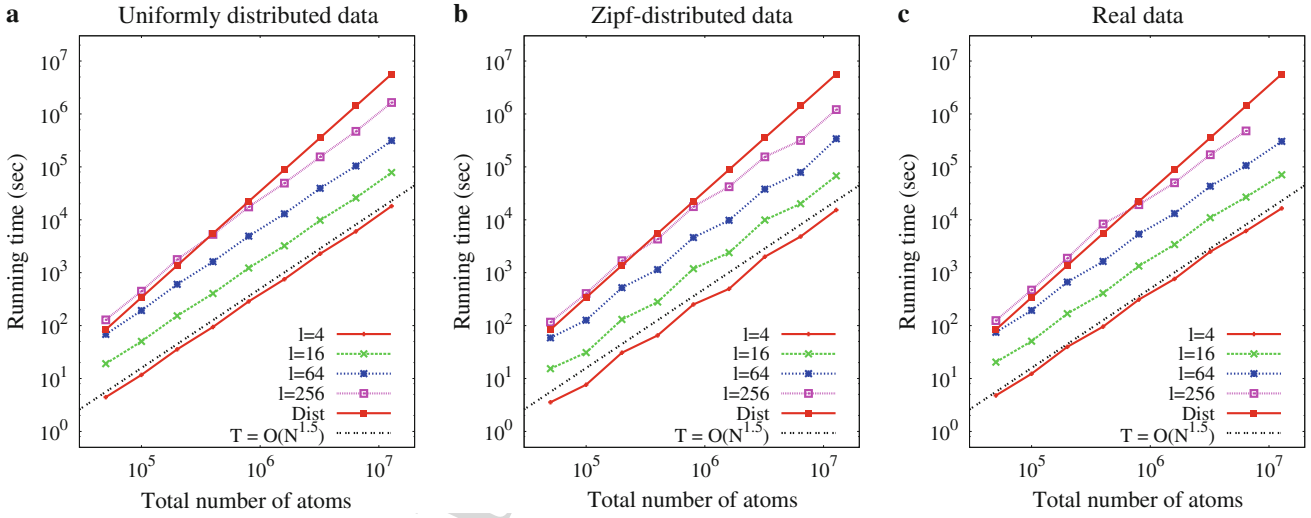


Fig. 18 Running time of the DT-SDH algorithm with 2D data

clusters in the data. We plot the running time of each run of DT- SDH as a dot. By comparing the results of the Zipf data to those of the uniform data, we can easily see that, at most of the time, the time spent to compute SDH in a Zipf dataset is less than that for the uniform dataset. The only exceptions are those generated from one random seed under Zipf order

Journal: 778 Article No.: 0205 MS Code: VLDB-D-09-00108.2 ☐ TYPESET ☐ DISK ☐ LE ☐ CP Disp.:2010/10/20 Pages: 24 Layout: Large

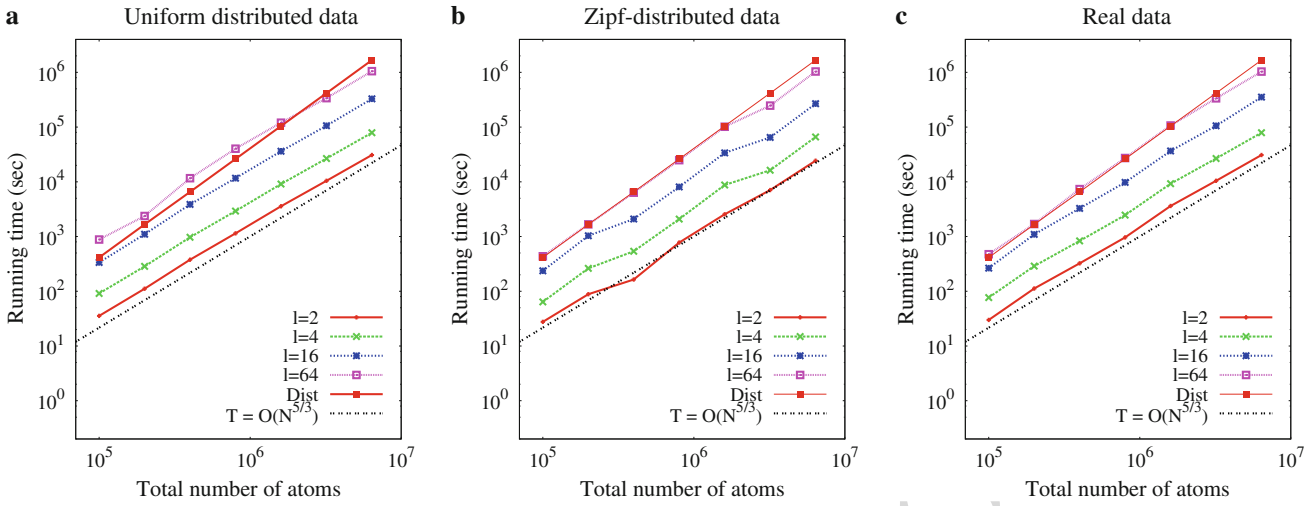


Fig. 19 Running time of the DT-SDH algorithm with 3D data

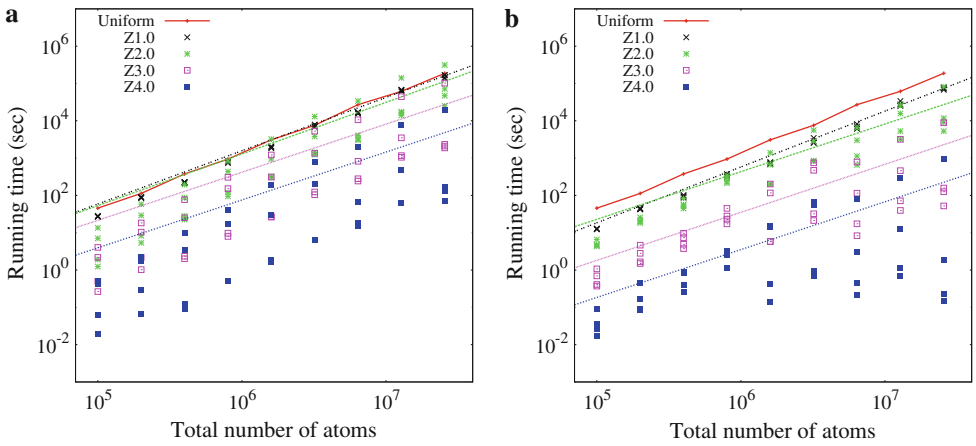


Fig. 20 Running time of DT- SDH with Zipf input data under different orders. Both *left* and *right* graphs are plotted on the exact same scale for easy comparisons

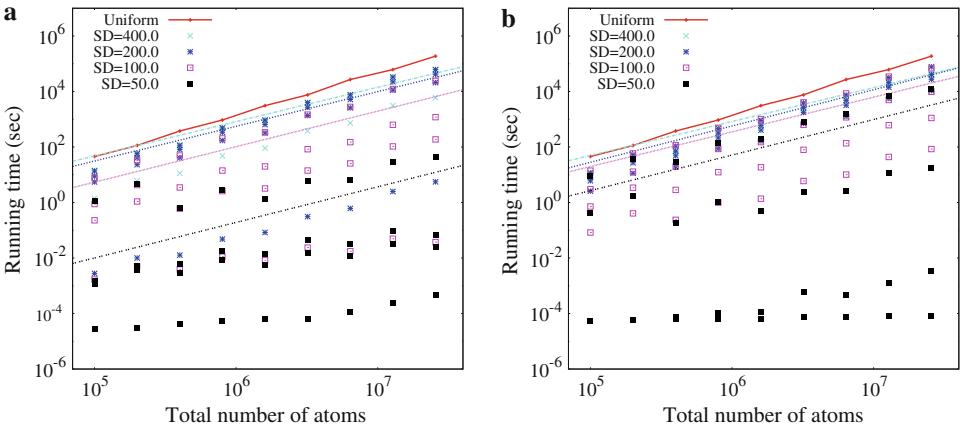
2.0 and block size 50 (Fig. 20a). We will scrutinize those cases later. When the Zipf order increases from 1.0 to 4.0, we can observe two trends:

(1) the running time decreases. In some cases of Zipf order 4.0, we can see a decrease of up to 4 orders of magnitude; and
(2) the variances of the running time among the four random datasets (under the same N) increase.

The first observation directly shows that data skewness has positive effects on the efficiency of DT- SDH in general. The large variances for the high-order Zipf cases indicate that the position of clusters plays a role in determining running time, given the fact that all four runs used data with the exact same "skewness". We also used the Gnuplot function-fitting tools to derive functions that describe the relationship between N and the running time for all Zipf orders. Specifically, we fit the dots into functions of the form $T = aN^b + c$ and such functions are drawn in the same color as that of their corresponding dots in Fig. 20. The positions of such lines in Fig. 20 show the above trends clearly. In Fig. 20a, the function of Zipf order 1.0 (e.g., 'Z1.0') has a similar slope (i.e, 1.427) to that of the uniform data (e.g., 1.5) while the slopes of higher-order Zipf datasets are in the range of (1.27, 1.28). This shows that, in addition to the absolute running time, the time complexity of DT- SDH also tends to decrease when more skewed data are input. One thing to point out is: nonlinear function fitting is not exact science and the details of the function-fitting methods used by Gnuplot are not revealed. Therefore, the parameter b in the fitted functions (i.e., slopes of the lines) can only be regarded as an indication of the algorithm's time complexity.

By decreasing the block size of the Zipf distribution, we will generate more "skewed" data. As a result (see Fig. 20b), we recorded shorter running times for almost all experimental runs when compared to those with block size 50. This can be easily captured by comparing the locations of corresponding

Journal: 778 Article No.: 0205 MS Code: VLDB-D-09-00108.2 TYPESET DISK LE CP Disp.:2010/10/20 Pages: 24 Layout: Large



Results of data with three clusters. Results of data with five clusters.

Fig. 21 Running time of DT- SDH with mixed-Gaussian data under different standard deviations. Both left and right graphs are plotted on the exact same scale for easy comparisons

dots and fitted functions in Fig. 20b and a. While the line slopes are still in the neighborhood of 1.28 for Zipf data with orders 2.0, 3.0, and 4.0, the a parameters of the fitted functions are of much smaller values than in Fig. 20a. In the case of Zipf with orders 3.0 and 4.0, a difference of more than one order of magnitude can be observed. Again, the slope of the Zipf 1.0 line (1.496) is close to that of uniform, confirming the results shown previously in Fig. 18.

Figure 21 shows the running time with the mixed-Gaussian data. The results are very similar to those in Fig. 20. When the SD decreases, the skewness of data increases, and the running time also decreases. In the extreme case of $SD = 50$, most datasets are processed within a fraction of a second (it went as low as 10^{-5} s). The variance of the running time among the four runs of each experiment also increases as SD becomes smaller. The fitted functions of all mixed-Gaussian experiments have slopes in the range of [1.22, 1.30], which is again significantly smaller than the 1.5 of the uniform data results. The data related to Fig. 21a were generated from a mixture of three Gaussian distributions while those in Fig. 21b mixture of five. The general trend is that the running time of experiments with the same parameters N and SD increases in Fig. 21b. Clearly, as the number of high-density data cluster increases (since each Gaussian gives rise to one cluster), the data become less skewed, and running time increases. In this set of experiments, we have seen no cases in which the mixed-Gaussian data required longer time to process than the corresponding uniform data. We believe the above results are another set of evidence that shows the benefits of skewed datasets increase as the data becomes more skewed.

In summary, our experiments show that DT- SDH is generally more efficient in processing skewed data. The more skewed the data is, the shorter the processing time is. In an extreme case in Fig. 20b, it takes only a fraction of a second to process a dataset with 25.6 million points! In addition to the absolute running time, we also believe the time complexity of DT- SDH can be lower than what we expect from Theorem 3 when the input data are very skewed.

The only "bad" cases (as shown in Fig. 20a) are caused by one random seed in generating Zipf data with order 2.0. By looking deeply into the actual data distributions in such cases, we found that there are 4 large clusters (ranked 3, 6, 7, and 8) falling into the non-coverable regions of the rank 1 cluster. As a result, distances are resolved in a lower rate than in the uniform data. On contrary to that, distances are consumed quickly in all other skewed datasets - we even observed several cases (for Zipf order 4.0) in which 100% of the distances are resolved. For the above inputs with excessively long processing time, we tested the remedy (2) introduced in Sect. 6.1.2 by computing a SDH with bucket width 9/10 of the original one. The results are very promising—the running time is reduced by up to three orders of magnitude (data not plotted).

8 Conclusions and future work

In this paper, we present analytical results related to the time complexity of a Quad-tree-based algorithm for computing many statistical measures of large-scale spatial data. The spatial distance histogram is one salient example of such measures. Being the main building blocks of high-level analytics in a wide range of computational science fields, such histograms are of great importance in domain-specific hypothesis testing and scientific discovery. This paper focuses on the methodology we adopt to accomplish the analysis: we transform the problem into quantifying the area of certain regions in space such that geometric modeling can be used to generate rigorous results. Our analysis shows that the algorithm has complexity $O(N^{\frac{3}{2}})$ for 2D data and $O(N^{\frac{5}{3}})$ for 3D data.

Journal: 778 Article No.: 0205 MS Code: VLDB-D-09-00108.2 TYPESET DISK LE CP Disp.:2010/10/20 Pages: 24 Layout: Large

To the best of our knowledge, this is the best result so far in the computation of exact SDH. We also show that the conclusion holds true under a wide range of spatial distributions of data points in the dataset, improving on previous conjectures that only consider uniformly distributed data.

Immediate future work in this area involves more explorations on the approximate algorithm, which is the main direction for developing practical fast solutions for SDH. While experimental results show very promising tradeoffs of running time and query error, probabilistic models have to be developed to study tight bounds of the error. Based on such models, more efficient and accurate heuristics for distributing distances into overlapping buckets can be designed. Eventually, the extension of our methodology to the computation of higher-order n-body correlation functions will depend on our explorations on the lower-order functions. Another direction is to compute the SDH in consecutive frames efficiently by taking advantage of the temporal locality of data points.

Acknowledgments The project described was supported by Award Number R01GM086707 from the National Institute Of General Medical Sciences (NIGMS) at the National Institutes of Health (NIH). The content is solely the responsibility of the authors and does not necessarily represent the official views of NIGMS or NIH.
The authors would also like to thank Dr. Sagar Pandit in the Department of Physics at the University of South Florida for discussions on general particle simulation problems.

Appendix

A The area of coverable region for $m > 1$ and $i \geq 2$

First, we get the magnitude of angle BCD by

$$\angle BCD = \angle DCE - \angle FCE = \arctan\frac{DE}{EC} - \frac{\pi}{4}$$
$$= \arctan\frac{\sqrt{[(i-1)p]^2 - \left(\frac{\delta}{2} - \frac{\delta}{2^m}\right)^2}}{\frac{\delta}{2} - \frac{\delta}{2^m}} - \frac{\pi}{4}$$

The area of the sector $\widehat{BDC}$ is $\frac{1}{2}[(i-1)p]^2\angle BCD$, and the area of the region $\widehat{BDGF}$ is

$$S_{\widehat{BDGF}} = S_{\widehat{BDC}} - S_{\triangle DHC} - S_{\triangle FGH}$$
$$= \frac{1}{2}[(i-1)p]^2\angle BCD - \frac{1}{2}EC(DE - HE) - \frac{\delta^2}{8}$$
$$= \frac{1}{2}[(i-1)p]^2\left[\arctan\frac{\sqrt{[(i-1)p]^2 - \delta^2\theta_m^2}}{\delta\theta_m} - \frac{\pi}{4}\right]$$
$$-\frac{\delta}{2}\theta_m\left[\sqrt{[(i-1)p]^2 - (\delta\theta_m)^2} - \delta\theta_m\right] - \frac{\delta^2}{8}$$

Finally, we get the area of the coverable region for $i \geq 2, m > 1$ as

$$S_{A'} = S_{out}(i) - 8S_{\widehat{BDGF}} - S_A$$
$$= \pi(ip)^2 + 4ip\left(\delta - \frac{2\delta}{2^m}\right) + \left(\delta - \frac{2\delta}{2^m}\right)^2$$
$$- 4[(i-1)p]^2\left[\arctan\frac{\sqrt{[(i-1)p]^2 - \delta^2\theta_m^2}}{\delta\theta_m} - \frac{\pi}{4}\right]$$
$$+ 4\delta\theta_m\left[\sqrt{[(i-1)p]^2 - (\delta\theta_m)^2} - \delta\theta_m\right] \quad (18)$$

B Volume of region B in 3D case

$$V_{\mathbf{B}} = \iint_{\mathbf{B}} \mathrm{d}x\mathrm{d}y \int_{\delta/2}^{\sqrt{p^2-x^2-y^2}} dz$$
$$= \iint_{\mathbf{B}} \left(\sqrt{p^2 - x^2 - y^2} - \frac{\delta}{2}\right)\mathrm{d}x\mathrm{d}y$$
$$= \int_a^{\frac{\pi}{4}} \mathrm{d}\theta \int_b^c \left(\sqrt{p^2 - r^2} - \frac{\delta}{2}\right) r\mathrm{d}r$$
$$= \int_a^{\frac{\pi}{4}} \left[-\frac{1}{3}(p^2 - r^2)^{\frac{3}{2}} - \frac{\delta}{4}r^2\right]\Big|_b^c d\theta$$
$$= \int_a^{\frac{\pi}{4}} \left[-\frac{\delta^3}{24} + \frac{1}{3}\left(p^2 - b^2\right)^{\frac{3}{2}} - \frac{\delta}{4}c^2 + \frac{1}{16}\frac{\delta^3}{(\sin\theta)^2}\right]\mathrm{d}\theta,$$

in which $a = \arctan\frac{\frac{\delta}{2}}{\sqrt{p^2 - 2\left(\frac{\delta}{2}\right)^2}}$, $c = \sqrt{p^2 - \left(\frac{\delta}{2}\right)^2}$, and $b = \frac{\delta}{2\sin\theta}$.

C The derivation of Eq. (11)

We accomplish this proof by studying the difference between $\frac{A(m)}{B(m)}$ and $\frac{1}{2}$. First, we see

$$A(m) - \frac{B(m)}{2} = 8\sum_{i=2}^{l}(i-1)^2\arctan\frac{\sqrt{8(i-1)^2 - \theta_{m+1}^2}}{\theta_{m+1}}$$
$$-4\sum_{i=2}^{l}\theta_{m+1}\sqrt{2(i-1)^2 - \theta_{m+1}^2}$$
$$+2\sum_{i=2}^{l}\theta_m\sqrt{2(i-1)^2 - \theta_m^2} + \sum_{i=2}^{l}\sqrt{2(i-1)^2 - \frac{1}{4}}$$

$$-4\sum_{i=2}^{l}(i-1)^2\arctan\frac{\sqrt{8(i-1)^2-\theta_m^2}}{\theta_m}$$
$$-4\sum_{i=2}^{l}(i-1)^2\arctan\sqrt{8(i-1)^2-1} \tag{19}$$

When $l\to\infty$, we have the following approximations:

$$\sum_{i=2}^{l}\sqrt{2(i-1)^2-\frac{1}{4}}\longrightarrow\sum_{i=2}^{l}\sqrt{2}(i-1),$$
$$\sum_{i=2}^{l}\theta_{m+1}\sqrt{2(i-1)^2-\theta_{m+1}^2}\longrightarrow\sum_{i=2}^{l}\theta_{m+1}\sqrt{2}(i-1)$$
$$\sum_{i=2}^{l}\theta_m\sqrt{2(i-1)^2-\theta_m^2}\longrightarrow\sum_{i=2}^{l}\theta_m\sqrt{2}(i-1),$$
$$\sum_{i=2}^{l}(i-1)^2\arctan\frac{\sqrt{8(i-1)^2-\theta_{m+1}^2}}{\theta_{m+1}}$$
$$\longrightarrow\sum_{i=2}^{l}(i-1)^2\arctan 2\sqrt{2}(i-1)$$
$$\sum_{i=2}^{l}(i-1)^2\arctan\frac{\sqrt{8(i-1)^2-\theta_m^2}}{\theta_m}$$
$$\longrightarrow\sum_{i=2}^{l}(i-1)^2\arctan 2\sqrt{2}(i-1)$$
$$\sum_{i=2}^{l}(i-1)^2\arctan\sqrt{8(i-1)^2-1}$$
$$\longrightarrow\sum_{i=2}^{l}(i-1)^2\arctan 2\sqrt{2}(i-1) \tag{20}$$

Plugging the left-hand side of six formulae in (20) into Eq. (19), we get $A(m)-\frac{B(m)}{2}\longrightarrow 0$ and thus $A(m)\longrightarrow\frac{B(m)}{2}$.

D Proof of Theorem 2

Proof Proof is accomplished in a similar way to that of Theorem 1. We have $\frac{\alpha(m+1,s)}{\alpha(m,s)}=\frac{A(m,s)}{B(m,s)}$ where

$$A(m,s)=1+\frac{4\sqrt{2}(l+l^2)}{s^{1+m}}-l\left(1-\frac{2}{s^{1+m}}\right)^2$$
$$+4(l-1)\left(\frac{1}{2}-\frac{1}{s^{1+m}}\right)^2$$
$$-4\sum_{i=2}^{l}\theta'_{m+1}\sqrt{2(i-1)^2-{\theta'_{m+1}}^2}$$
$$+8\sum_{i=2}^{l}(i-1)^2\arctan\frac{\sqrt{2(i-1)^2-{\theta'_{m+1}}^2}}{\theta'_{m+1}}$$
$$+\sum_{i=2}^{l}\sqrt{8(i-1)^2-1}$$
$$-8\sum_{i=2}^{l}(i-1)^2\arctan\sqrt{8(i-1)^2-1}, \tag{21}$$

and

$$B(m,s)=1+\frac{4\sqrt{2}(l+l^2)}{s^m}-l\left(1-\frac{2}{s^m}\right)^2$$
$$+4(l-1)\left(\frac{1}{2}-\frac{1}{s^m}\right)^2$$
$$-4\sum_{i=2}^{l}\theta'_m\sqrt{2(i-1)^2-{\theta'_m}^2}$$
$$+8\sum_{i=2}^{l}(i-1)^2\arctan\frac{\sqrt{2(i-1)^2-{\theta'_m}^2}}{\theta'_m}$$
$$+\sum_{i=2}^{l}\sqrt{8(i-1)^2-1}$$
$$-8\sum_{i=2}^{l}(i-1)^2\arctan\sqrt{8(i-1)^2-1} \tag{22}$$

As in Appendix C, by comparing the value of $\frac{A(m,s)}{B(m,s)}$ to $\frac{1}{s}$, we get

$$A(m,s)s-B(m,s)=(s-1)\sum_{i=2}^{l}\sqrt{8(i-1)^2-1}$$
$$-8(1-s)\sum_{i=2}^{l}(i-1)^2\arctan\sqrt{8(i-1)^2-1}$$
$$-4(1-s)\sum_{i=2}^{l}\theta'_{m+1}\sqrt{2(i-1)^2-{\theta'_{m+1}}^2}$$
$$+8(s-1)\sum_{i=2}^{l}(i-1)^2\arctan\frac{\sqrt{2(i-1)^2-{\theta'_{m+1}}^2}}{\theta'_{m+1}} \tag{23}$$

When $l\to\infty$, we have the following approximations.

$$\sum_{i=2}^{l}\sqrt{2(i-1)^2-{\theta'_{m+1}}^2}\longrightarrow\frac{1}{2}\sum_{i=2}^{l}\sqrt{8(i-1)^2-1},$$
$$\sum_{i=2}^{l}(i-1)^2\arctan\frac{\sqrt{8(i-1)^2-{\theta'_{m+1}}^2}}{\theta'_{m+1}}$$
$$\longrightarrow\sum_{i=2}^{l}(i-1)^2\arctan\sqrt{8(i-1)^2-1} \tag{24}$$

Journal: 778 Article No.: 0205 MS Code: VLDB-D-09-00108.2 TYPESET DISK LE CP Disp.:2010/10/20 Pages: 24 Layout: Large

Plugging the left-hand side of the above two formulae into Eq. (23), we get $sA(m,s) - B(m,s) \longrightarrow 0$ and this concludes the proof.

E Quantities related to Theorem 4

For easy presentation, we denote $x = r/\delta$. The maximal bucket region for the first bucket is

$$g(1) = \pi q^2 + 4q\delta + \delta^2$$

and that for the second bucket is

$$g(2) = \left\{\pi(\sqrt{2}+x)^2 + 4(\sqrt{2}+x) + 1 - 8\left[\left(\arctan\sqrt{7} - \frac{\pi}{4}\right) - \frac{1}{8}(\sqrt{7}-1)\right]\right\}\delta^2$$

The coverable region for bucket 1 is

$$f(1,m) = \left[2\pi + 4\sqrt{2}\left(1 - \frac{2}{2^m}\right) - \frac{4}{2^m} + \frac{4}{2^{2m}} + 1\right]\delta^2$$

and that for bucket 2 is

$$f(2,m) = \left\{\pi(\sqrt{2}+x)^2 + 4(\sqrt{2}+x)\left(1 - \frac{2}{2^m}\right) - \frac{4}{2^m} + \frac{4}{2^{2m}} + 1 - 8\left\{\left[\arctan\frac{\sqrt{2-\theta_m^2}}{\theta_m} - \frac{\pi}{4}\right] - \frac{1}{2}\left[\sqrt{2-\theta_m^2} - \theta_m\right]\theta_m\right\}\right\}\delta^2$$

Therefore, we have $\frac{\alpha(m+1)}{\alpha(m)} = \frac{A(m)}{B(m)}$ where

$$A(m) = 4(2\sqrt{2}+x)\frac{2}{2^{m+1}} + \frac{8}{2^{m+1}} - \frac{8}{2^{2m+2}} + \sqrt{7} - 1 - 8\arctan\sqrt{7} + 8\arctan\frac{\sqrt{2-\theta_{m+1}^2}}{\theta_{m+1}} - 4\left[\sqrt{2-\theta_{m+1}^2} - \theta_{m+1}\right]\theta_{m+1}$$

and

$$B(m) = 4(2\sqrt{2}+x)\frac{2}{2^m} + \frac{8}{2^m} - \frac{8}{2^{2m}} + \sqrt{7} - 1 - 8\arctan\sqrt{7} + 8\arctan\frac{\sqrt{2-\theta_m^2}}{\theta_m} - 4\left[\sqrt{2-\theta_m^2} - \theta_m\right]\theta_m$$

In a straightforward way, the above can give rise to the following.

$$A(m) \le \frac{1}{2}B(m)$$

References

1. Allen, M.: Introduction to Molecular Dynamics Simulation. John von Neumann Institute of Computing, NIC Seris, vol. 23 (2003)
2. Allen, M.P., Tildesley, D.J.: Computer Simulations of Liquids. Clarendon Press, Oxford (1987)
3. Arya, M., Cody, W.F., Faloutsos, C., Richardson, J., Toya, A.: QBISM: Extending a DBMS to Support 3D Medical Images. In: ICDE, pp. 314–325, (1994)
4. Bamdad, M., Alavi, S., Najafi, B., Keshavarzi, E.: A new expression for radial distribution function and infinite shear modulus of lennard-jones fluids. Chem. Phys. **325**, 554–562 (2006)
5. Barnes, J., Hut, P.: A hierarchical O(N log N) force-calculation algorithm. Nature **324**(4), 446–449 (1986)
6. Brown, P.G.: Overview of scidb: large scale array storage, processing and analysis. In: SIGMOD Conference, pp. 963–968 (2010)
7. Callahan, P.B., Kosaraju, S.R.: A decomposition of multidimensional point sets with applications to k-nearest-neighbors and n-body potential fields. J. ACM **42**(1), 67–90 (1995)
8. Cormen, T.H., Leiserson, C.E., Rivest, R.L., Stein, C.: Introduction to Algorithms. pp. 73–75. 2nd edn. MIT Press and McGraw-Hill, Cambridge (2001)
9. Csabai, I., Trencseni, M., Dobos, L., Jozsa, P., Herczegh, G., Purger, N., Budavari, T., Szalay, A.S.: Spatial indexing of large multidimensional databases. In: Proceedings of the 3rd Biennial Conference on Innovative Data Systems Resarch (CIDR), pp. 207–218 (2007)
10. Eltabakh, M.Y., Ouzzani, M., Aref, W.G.: BDBMS—a database management system for biological data. In: Proceedings of the 3rd Biennial Conference on Innovative Data Systems Resarch (CIDR), pp. 196–206 (2007)
11. Feig, M., Abdullah, M., Johnsson, L., Pettitt, B.M.: Large scale distributed data repository: design of a molecular dynamics trajectory database. Future Gener. Comput. Syst. **16**(1), 101–110 (1999)
12. Filipponi, A.: The radial distribution function probed by X-ray absorption spectroscopy. J. Phys. Condens. Matt. **6**, 8415–8427 (1994)
13. Finocchiaro, G., Wang, T., Hoffmann, R., Gonzalez, A., Wade, R.: DSMM: a database of simulated molecular motions. Nucl. Acids Res. **31**(1), 456–457 (2003)
14. Frenkel, D., Smit, B.: Understanding Molecular Simulation: From Algorithm to Applications, volume 1 of Computational Science Series. Academic Press, New York (2002)
15. Gawlick, D., Lenkov, D., Yalamanchi, A., Chernobrod, L.: Applications for expression data in relational database system. In: ICDE, pp. 609–620 (2004)
16. Gray, A.G., Moore, A.W.: N-body problems in statistical learning. In: Advances in Neural Information Processing Systems (NIPS), pp. 521–527, MIT Press (2000)
17. Gray, J., Liu, D., Nieto-Santisteban, M., Szalay, A., DeWitt, D., Heber, G.: Scientific data management in the coming decade. SIGMOD Rec. **34**(4), 34–41 (2005)
18. Greengard, L., Rokhlin, V.: A fast algorithm for particle simulations. J. Comput. Phys. **135**(12), 280–292 (1987)
19. Heber, G., Gray, J.: Supporting finite element analysis with a relational database backend. Part I: there is life beyond files. Technical Report MSR-TR-2005-49, Microsoft Research (2005)
20. Hess, B., Kutzner, C., Spoel, D.van der , Lindahl, E.: GROMACS 4: algorithms for highly efficient, load-balanced, and scalable molecular simulation. J. Chem. Theory Comput. **4**(3), 435–447 (2008)
21. Howe, B., Maier, D., Bright, L.: Smoothing the ROI curve for scientific data management applications. In: CIDR, pp. 185–195 (2007)

22. Klasky, S., Ludaescher, B., Parashar, M.: The Center for Plasma Edge Simulation Workflow Requirements. In: EEE Workshop on Workflow and Data Flow for Scientific Applications (SciFlow'06), pp. 73–73 (1991)
23. Krishnamurthy, L., Nadeau, J., Ozsoyoglu, G., Ozsoyoglu, M., Schaeffer, G., Tasan, M., Xu, W.: Pathways database system: an integrated system for biological pathways. Bioinformatics **19**(8), 930–937 (2003)
24. Ma, X., Winslett, M., Norris, J., Jiao, X., Fiedler, R.: Godiva: lightweight data management for scientific visualization applications. In: ICDE, pp. 732–744 (2004)
25. Moore, A.W., Connolly, A.J., Genovese, C., Gray, A., Grone, L., Kanidoris, N. II., Nichol, R.C., Schneider, J., Szalay, A.S., Szapudi, I., Wasserman, L.: Mining the Sky, volume 2001 of ESO Astrophysics Symposia, Chapter Fast Algorithms and Efficient Statistics: N-Point Correlation Functions. pp. 71–82. Springer, Heidelberg (2006)
26. Omeltchenko, A., Campbell, T.J., Kalia, R.K., Liu, X., Nakano, A., Vashishta, P.: Scalable I/O of large-scale molecular dynamics simulations: a data-compression algorithm. Comput. Phys. Commun. **131**, 78–85 (2000)
27. Orenstein, J.A.: Multidimensional tries used for associative searching. Inf. Process. Lett. **14**(4), 150–157 (1982)
28. Patel, J.M.: The role of declarative querying in bioinformatics. OMICS J. Integr. Biol. **7**(1), 89–91 (2003)
29. Samet, H.: The quadtree and related hierarchical data structures. ACM Comput. Surv. **16**(2), 187–260 (1984)
30. Springel, V., White, S.D.M., Jenkins, A., Frenk, C.S., Yoshida, N., Gao, L., Navarro, J., Thacker, R., Croton, D., Helly, J., Peacock, J.A., Cole, S., Thomas, P., Couchman, H., Evrard, A., Colberg, J., Pearce, F.: Simulations of the formation, evolution and clustering of galaxies and quasars. Nature **435**, 629–636 (2005)
31. Stark, J.L., Murtagh, F.: Astronomical Image and Data Analysis. Springer, Heidelberg (2002)
32. Stonebraker, M., Madden, S., Abadi, D.J., Harizopoulos, S., Hachem, N., Helland, P.: The End of an Architectural Era (It's Time for a Complete Rewrite). In: VLDB, pp. 1150–1160 (2007)
33. Szalay, A.S., Gray, J., Thakar, A., Kunszt, P.Z., Malik, T., Raddick, J., Stoughton, C., vandenBerg, J.: The SDSS Skyserver: public access to the sloan digital sky server data. In: Proceedings of International Conference on Management of Data (SIGMOD), pp. 570–581 (2002)
34. Szapudi, I.: A new method for calculating counts in cells. Astrophys. J. **493**(1), 39–51 (1998)
35. Szapudi, I., Colombi, S., Bernardeau, F.: Cosmic statistics of statistics. Mon. Notes Roy. Astron. Soc. **310**(2), 428–444 (1999)
36. Tao, Y., Sun, J., Papadias, D.: Analysis of predictive spatio-temporal queries. ACM Trans. Database Syst. **28**(4), 295–336 (2003)
37. Tu, Y.-C., Chen, S., Pandit, S.: Computing Spatial Distance Histograms Efficiently in Scientific Databases. Technical Report CSE/08-103, http://www.cse.usf.edu/~ytu/pub/tr/pdh.pdf, Department of Computer Science and Engineering, University of South Florida (2008)
38. Tu, Y.-C., Chen, S., Pandit, S.: Computing distance histograms efficiently in scientific databases. In: Proceedings of International Conference on Data Engineering (ICDE), pp. 796–807 (2009)
39. Türker, C., Akal, F., Joho, D., Schlapbach, R.: B-fabric: an open source life sciences data management system. In: SSDBM, pp. 185–190 (2009)
40. Xu, W., Ozer, S., Gutell, R.R.: Covariant evolutionary event analysis for base interaction prediction using a relational database management system for RNA. In: SSDBM, pp. 200–216 (2009)

Journal: **778** Article No.: **0205** MS Code: **VLDB-D-09-00108.2** ☐ TYPESET ☐ DISK ☐ LE ☐ CP Disp.:**2010/10/20** Pages: **24** Layout: **Large**