Approximate Algorithms for Computing Spatial Distance Histograms with Accuracy Guarantees

Vladimir Grupcev, *Student Member*, *IEEE*, Yongke Yuan, Yi-Cheng Tu, *Member*, *IEEE*, Jin Huang, *Student Member*, *IEEE*, Shaoping Chen, Sagar Pandit, and Michael Weng

Abstract—Particle simulation has become an important research tool in many scientific and engineering fields. Data generated by such simulations impose great challenges to database storage and query processing. One of the queries against particle simulation data, the spatial distance histogram (SDH) query, is the building block of many high-level analytics, and requires quadratic time to compute using a straightforward algorithm. Previous work has developed efficient algorithms that compute exact SDHs. While beating the naive solution, such algorithms are still not practical in processing SDH queries against large-scale simulation data. In this paper, we take a different path to tackle this problem by focusing on approximate algorithms with provable error bounds. We first present a solution derived from the aforementioned exact SDH algorithm, and this solution has running time that is unrelated to the system size N. We also develop a mathematical model to analyze the mechanism that leads to errors in the basic approximate algorithm. Our model provides insights on how the algorithm can be improved to achieve higher accuracy and efficiency. Such insights give rise to a new approximate algorithm with improved time/accuracy tradeoff. Experimental results confirm our analysis.

Index Terms—Molecular simulation, spatial distance histogram, quadtree, scientific databases

1 Introduction

Many scientific fields have undergone a transition to data/computation intensive science, as the result of automated experimental equipments and computer simulations. In recent years, much progress has been made in building data management tools suitable for processing scientific data [1], [2], [3], [4], [5]. Scientific data impose great challenges to the design of database management systems that are traditionally optimized toward handling business applications. First, scientific data often come in large volumes that require us to rethink the storage, retrieval, and replication techniques in current DBMSs. Second, user accesses to scientific databases are focused on complex high-level analytics and reasoning that go beyond simple aggregate queries. While many types of domain-specific analytical queries are seen in scientific databases, the DBMS should support efficient processing of those that are frequently used as building blocks for more complex analysis. However, many of such basic analytical queries need superlinear processing time if handled in a straightforward way, as in current scientific databases. In this paper, we report our efforts to design efficient algorithms for a type of query that is extremely important in the analysis of *particle simulation data*.

Particle simulations are computer simulations in which the basic components (e.g., atoms, stars, etc.) of large systems (e.g., molecules, galaxies, etc.) are treated as classical entities that interact for certain duration under postulated empirical forces. For example, molecular simulations (MSs) explore relationship between molecular structure, movement, and function. These techniques are primarily applicable in modeling of complex chemical and biological systems that are beyond the scope of theoretical models. MS has become an important research tool in material sciences [6], astrophysics [7], biomedical sciences, and biophysics [8], motivated by a wide range of applications. In astrophysics, the N-body simulations are predominantly used to describe large scale celestial structure formation [8], [9], [10], [11]. Similar to MS in applicability and simulation techniques, the N-body simulation comes with even larger scales in terms of total number of particles simulated.

Results of particle simulations form large data sets of particle configurations. Typically, these configurations store information about the particle types, their coordinates, and velocities—the same type of data we have seen in spatial-temporal databases [12]. While snapshots of configurations are interesting, quantitative structural analysis of interatomic structures are the mainstream tasks in data analysis. This requires the calculation of statistical properties or functions of particle coordinates [9]. Of special interest to scientists are those quantities that require coordinates of

- *V. Grupcev and Y.-C. Tu are with the Department of Computer Science and Engineering, University of South Florida, 4202 E. Fowler Ave., ENB 118, Tampa, FL 33620. E-mail: vgrupcev@mail.usf.edu, ytu@cse.usf.edu.*
- *Y. Yuan and M. Weng are with the Department of Industrial and Management Systems Engineering, University of South Florida, 4202 E. Fowler Ave., ENB118, Tampa, FL 33620. E-mail: {yyuan, mxweng}@usf.edu.*
- *J. Huang is with the Department of Computer Science, University of Texas at Arlington, 500 UTA Boulevard, Room 640, ERB Buildings, Arlington, TX 76019. E-mail: jin.huang@mavs.uta.edu.*
- *S. Chen is with the Department of Mathematics, Wuhan University of Technology, 122 Luosi Road, Wuhan, Hubei 430070, P.R. China. E-mail: chensp@whut.edu.cn.*
- *S. Pandit is with the Department of Physics, University of South Florida, 4202 E. Fowler Ave., PHY114, Tampa, FL 33620. E-mail: pandit@cas.usf.edu.*

Manuscript received 18 Oct. 2011; revised 16 Apr. 2012; accepted 12 July 2012; published online 25 July 2012.
Recommended for acceptance by C. Shahabi.
For information on obtaining reprints of this article, please send e-mail to: tkde@computer.org, and reference IEEECS Log Number TKDE-2011-10-0639.
Digital Object Identifier no. 10.1109/TKDE.2012.149.

two particles simultaneously. In their brute-force form, these quantities require $O(N^2)$ computations for N particles [8]. In this paper, we focus on one such analytical query: the *Spatial Distance Histogram (SDH)* query, which asks for a histogram of the distances of all pairs of particles in the simulated system.

1.1 Problem Statement

The problem can be defined as follows: given the coordinates of N points in space, we are to compute the counts of point-to-point distances that fall into a series of l ranges in the $\mathbb{R}$ domain: $[r_0, r_1), [r_1, r_2), [r_2, r_3), \dots, [r_{l-1}, r_l]$. A range $[r_i, r_{i+1})$ in such series is called a *bucket*, and the span of the range $r_{i+1} - r_i$ is called the *width* of the bucket. In this paper, we focus our discussions on the case of *standard SDH queries*, where all buckets have the same width p and $r_0 = 0$, which gives the following series of buckets: $[0, p), [p, 2p), \dots, [(l-1)p, lp]$. Generally, the boundary of the last bucket lp is set to be the maximum distance of any pair of points in the data set. Although almost all scientific data analysis only require the computation of standard SDH queries, our solutions can be easily extended to handle histograms with nonuniform bucket width and/or arbitrary values of r_0 and r_l.[1] The answer to an SDH query is basically a series of nonnegative integers $\mathbf{h} = (h_1, h_2, \dots, h_l)$, where h_i $(0 < i \le l)$ is the number of pairs of points whose distances are within the bucket $[(i-1)p, ip)$.

1.2 Motivation

The SDH is a fundamental tool in the validation and analysis of particle simulation data. It serves as the main building block of a series of critical quantities to describe a physical system. Specifically, SDH is a direct estimation of a continuous statistical distribution function called *radial distribution functions* (RDF) [7], [9], [13] that is defined as

$$g(r) = \frac{N(r)}{4\pi r^2 \delta r \rho}, \tag{1}$$

where $N(r)$ is the expected number of atoms in the shell between r and $r + \delta r$ around any particle, ρ is the average density of particles in the whole system, and $4\pi r^2 \delta r$ is the volume of the shell. Since SDH directly provides the value for $N(r)$, the RDF can be viewed as a normalized SDH.

The RDF is of great importance in computation of thermodynamic quantities about the simulated system. Some of the important quantities like total pressure,

$$p = \rho kT - \frac{2\pi}{3}\rho^2 \int dr r^3 u'(r) g(r, \rho, T),$$

and energy,

$$\frac{E}{NkT} = \frac{3}{2} + \frac{\rho}{2kT}\int dr\ 4\pi r^2 u(r) g(r, \rho, T),$$

can be derived in terms of structure factor that can be expressed using $g(r)$ [14]. For monoatomic systems, the relation between RDF and the structure factor of the system [15] takes simple form, viz.,

$$S(k) = 1 + \frac{4\pi\rho}{k}\int_0^\infty (g(r) - 1) r \sin(kr)\, dr.$$

The definitions of all notations in the above formulas can be found in [14] and [15]. To compute SDH in a straightforward way, we have to calculate distances between all pairs of particles and put the distances into bins with a user-specified width, as done in state-of-the-art simulation data analysis software packages [7], [16]. MS or N-body techniques generally consist of large number of particles. For example, the Virgo consortium has accomplished a simulation containing 10 billion particles to study the formation of galaxies and quasars [17]. This kind of scale prohibits the analysis of large data sets following the brute-force approach. From a database viewpoint, it would be desirable to make SDH a basic query type with the support of scalable algorithms.

Previous work [18], [19] have addressed this problem by developing algorithms that compute exact SDHs with time complexity lower than quadratic. The main idea is to organize the data in a space-partitioning tree and process pairs of tree nodes instead of pairs of particles (thus saving processing time). The tree structure used include kd-tree in [18] and region quad/oct-tree in our previous work [19], which also proved that the time complexity of such algorithms is $O(N^{\frac{2d-1}{d}})$, where $d \in \{2, 3\}$ is the number of dimensions in the data space. While beating the naive solution in performance, such algorithms' running time for large data sets can still be undesirably long. On the other hand, an SDH with some bounded error can satisfy the needs of users. In fact, there are cases where even a coarse SDH will greatly help the fine-tuning of simulation programs [9]. Generally speaking, the main motivation to process SDHs is to study the statistical distribution of point-to-point distances in the simulated system [9]. Since a histogram by itself is an approximation of the underlying distribution $g(r)$ (1), an inaccurate histogram generated from a given data set will still be useful in a statistical sense. Therefore, in this paper, we focus on approximate algorithms with very high performance that deliver query results with low error rates. In addition to experimental results, we also evaluate the performance/accuracy tradeoffs provided by the proposed algorithms in an analytical way. The running time of our proposed algorithm is only related to the desired accuracy. Our experimental results show significant improvement in performance/accuracy tradeoff of our algorithm over the previous algorithms—the error rates in query results are very small even when the running time is reasonably short.

1.3 Roadmap of the Paper

We continue this paper by a survey of related work and a list of our contributions in Section 2; we introduce the technical background on which our approximate algorithm is built in Section 3; we describe the details of a basic approximate algorithm and relevant empirical evaluation in Section 4; we dedicate Section 6 to mathematical analysis of the key mechanisms in our basic algorithm; the results of our analytical work are used to develop a new algorithm with improved performance and we introduce and evaluate that algorithm in Section 7; Section 8 concludes this paper.

1. The only complication of nonuniform bucket width is that, given a distance value, we need $O(\log l)$ time to locate the bucket instead of constant time for equal bucket width.

2 Related Work and Our Contributions

The scientific community has gradually moved from processing large data files toward using database systems for the storage, retrieval, and analysis of large-scale scientific data [2], [20]. Conventional (relational) database systems are designed and optimized toward data and applications from the business world. In recent years, the database community has invested much efforts into constructing database systems that are suitable for handling scientific data. For example, the BDBMS project [3] handles annotation and provenance of biological sequence data; and the PeriScope [5] project is aimed at efficient processing of declarative queries against biological sequences. In addition to that, there are also proposals of new DBMS architectures for scientific data management [21], [22], [23]. The main challenges and possible solutions of scientific data management are discussed in [1].

Traditionally, MS data are stored in large files and queries are implemented in stand-alone programs, as represented by popular simulation/analytics packages [16]. Recent efforts have been dedicated to building simulation data management systems on top of relational databases, as represented by the BioSimGrid [4] and SimDB [24] projects developed for MSs. However, such systems are still in short of efficient query processing strategies. To the best of our knowledge, the computation of SDH in such software packages is done in a brute-force way, which requires $O(N^2)$ time.

In particle simulations, the computation of (gravitational/electrostatic) force is similar to the SDH problem. Specifically, the force is the sum of all pairwise interactions in the system, thus requires $O(N^2)$ steps to compute. The simulation community has adopted approximate solutions represented by the Barnes-Hut algorithm that runs on $O(N \log N)$ time [25] and the Multipole algorithm [26] with linear running time. Although all above algorithms use a tree-like data structure to hold the data, they provide little insights on how to solve the SDH problem. The main reason is that these strategies take advantage of two features of force: 1) for any pairwise interaction, its contribution to the force decreases dramatically when particle distance increases; 2) the effects of symmetric interactions cancel out. However, neither features are applicable to SDH computation, in which every pairwise interaction counts and all are equally important. Another method for force computation is based on well-separated pair decomposition (WSPD) [27] and was found to be equivalent to the Barnes-Hut algorithm. A WSPD is a collection of pairs of subsets of the data such that all point-to-point distances are covered by such pairs. The pairs of subsets are also well separated in that the smallest distance between the smallest balls covering the subsets (with radius r) is at least sr, where s is a user-defined parameter. Although relevant by intuition, the WSPD does not produce fast solution for SDH computation.

It is worth mentioning that there has been work done on a broader problem of histogram computation in the context of data stream management [28]. The data stream systems usually work with distributive aggregates [28] such as COUNT, SUM, MAX, and MIN which may be computed incrementally using constant space and time. They also tackle so called holistic aggregates such as TOP-k [29], [30], QUANTILE [31], and COUNT DISTINCT [32], [33]. When computing the holistic aggregates they have utilized hash-based functions that produce histograms [30], [34]. But the data stream community has never specifically worked on the problem of computing a histogram that will disclose the distance counts belonging to a particular range (a bucket), i.e., an SDH. After thoroughly reviewing their work, we believe that none of their proposed solutions is directly applicable to the problem of SDH computation stated in this paper.

Another similar problem to the SDH computation is to find k-nearest neighbors (kNN) in a high-dimensional space [35]. In such a problem, avoidance of distance computation is the primary goal in algorithmic design due to the high cost of such operations. The main technique is to choose a set of reference points (i.e., pivots) in the database and precompute distances between data points to the pivots. In processing kNN queries, the search space can be pruned based on the precomputed distances. However, being a searching problem, kNN is very different from the counting-based SDH problem. As a result, the data structures and algorithmic details shown in [35] have little overlap with our solutions to the SDH problem.

Although SDH is an important analytics, there is not much elaboration on efficient SDH algorithms. An earlier work from the data mining community [18] opened the direction of processing SDHs by space-partitioning trees. The core idea is to process all the particles in a tree node as one single entity to take advantage of the nonzero bucket width p. By this, processing time is saved by avoiding computation of particle-to-particle distances. Our earlier paper [19] proposed a similar algorithm as well as rigorous mathematical analysis (not found in [18]) of the algorithm's time complexity. Specifically, in [19], we proposed a novel algorithm (named *DM-SDH*) to compute SDH based on a data structure called *density map*, which can be easily implemented by augmenting a Quadtree index. Contrary to that, the data structure adapted in [18] is the kd-tree. Our mathematical analysis [36] has shown that the algorithm runs on $\Theta(N^{\frac{3}{2}})$ for two-dimensional data and $\Theta(N^{\frac{5}{3}})$ for three-dimensional data, respectively. The technical details of such an algorithm will be introduced in Section 3.

This paper significantly extends our earlier work [19] by focusing on approximate algorithms for SDH processing. In particular, we claim the following contributions via this work:

1. We present an approximate SDH processing strategy that is derived from the basic exact algorithm, and this approximate algorithm has constant-time complexity and a provable error bound;
2. We develop a mathematical model to analyze the effects of error compensation that led to high accuracy of our algorithm; and
3. We propose an improved approximate algorithm based on the insights obtained from the above analytical results.

It is also worth mentioning that we have recently published another paper [37] in this field. That paper

TABLE 1
Symbols and Notations

Symbol	Definition
p	width of histogram buckets
l	total number of histogram buckets
$\mathbf{h}$	the histogram with elements h_i $(0 < i \leq l)$
N	total number of particles in data
i	an index symbol for any series
DM_i	the I-th level density map
d	number of dimensions of data
ϵ	error bound for the approximate algorithm
H	total level of density maps, i.e., tree height

focuses on a more sophisticated heuristics to generate approximate results based on spatial uniformity of data items. Such heuristics improves the accuracy of each distance distribution, which is the basic operation of our approximate algorithm (Section 4.1). In this paper, we introduce the design of the approximate algorithm and its performance analysis. Technically, we emphasize the impacts of error compensation among different distribution operations. As a result, we do not require low error rates to be obtained from each operation, as we show the total error is low even when a primitive heuristics is used. In other words, these two papers, although both take approximate SDH processing as the basic theme, make their contributions at two different levels of the problem. Work in [37] focuses on improving accuracy of single distribution operations while this paper, in addition to a systematic description of the algorithm, studies how errors from different distribution operations cancel out each other, and to what extent such error compensation affects the accuracy of the final results.

3 Preliminaries

In this section, we introduce the algorithm we developed in [19] to compute exact SDHs. Techniques and analysis related to this algorithm are the basis for the approximate algorithm we focus on in this paper. In Table 1, we list the notations that are used throughout this paper. Note that symbols defined and referenced in a local context are not listed here.

3.1 Overview of the Density Map-Based SDH (DM-SDH) Algorithm

To beat the $O(N^2)$ time needed by the naive solution, we need to avoid the computation of all particle-to-particle distances. An important observation here is: a histogram bucket always has a nonzero width p. Given a pair of points, their bucket membership could be determined if we only know a range that the distance belongs to and this range is contained in a histogram bucket. The central idea of our approach is a data structure called *density map*, which is basically a grid containing cells of equal size.[2] In every cell of the grid, we record the number of particles that are located in the space represented by that cell as well as the

2. From now on, we use 2D data and grids to elaborate our ideas unless specified otherwise. Note that extending our discussions to 3D data/space would be straightforward, as studied in [36].

```
Algorithm DM-SDH
Inputs: all data points, density maps built beforehand,
and bucket width $p$
Output: an array of counts $\mathbf{h}$

1  initialize all elements in $\mathbf{h}$ to 0
2  find the first density map $DM_o$ with cells diagonal
   length $k \leq p$
3  for all cells in $DM_o$
4  do $n \leftarrow$ number of particles in the cell
5      $h_0 \leftarrow h_0 + \frac{1}{2}n(n-1)$
6  for any two cells $M_j$ and $M_k$ in $DM_o$
7  do RESOLVETWOCELLS ($M_j$, $M_k$)
8  return $\mathbf{h}$

Procedure RESOLVETWOCELLS ($M_1$, $M_2$)
0  check if $M_1$ and $M_2$ are resolvable
1  if $M_1$ and $M_2$ are resolvable
2  then $i \leftarrow$ index of the bucket $M_1$ & $M_2$ resolve into
3       $n_1 \leftarrow$ number of particles in $M_1$
4       $n_2 \leftarrow$ number of particles in $M_2$
5       $h_i \leftarrow h_i + n_1 n_2$
6  else if $M_1$ & $M_2$ are on the last density map
7       for each particle A in $M_1$
8           for each particle B in $M_2$
9           do $f \leftarrow$ distance between A and B
10              $i \leftarrow$ the bucket $f$ falls into
11              $h_i \leftarrow h_i + 1$
12 else
13      $DM' \leftarrow$ next density map with higher resolution
14      for each partition $M_1'$ of $M_1$ on $DM'$
15          for each partition $M_2'$ of $M_2$ on $DM'$
16          do RESOLVETWOCELLS ($M_1'$, $M_2'$)
```

Fig. 1. The DM-SDH algorithm.

four coordinates that determine the exact boundary of the cell in space. The reciprocal of the cell size in a density map is called the *resolution* of the density map. To process the SDH query, we build a series of density maps with different resolutions. We organize all such density maps into a point region (PR) Quadtree [38], in which the resolution of a density map (i.e., all nodes on level i of the tree) is always doubled as compared to the previous one (i.e., those on level $i-1$) in the series.

The pseudocode of the DM-SDH algorithm can be found in Fig. 1. The core of the algorithm is a procedure named RESOLVETWOCELLS, which is given as input a pair of cells M_1 and M_2 on the same density map. In RESOLVETWOCELLS, we first compute the minimum and maximum distances between any particle from M_1 and any one from M_2 (line 1). Obviously, this can be accomplished in constant time given the corner coordinates of two cells stored in the density map. When the minimum and maximum distances between M_1 and M_2 fall into the same histogram bucket i, we say these two cells are *resolvable* on this density map, and they *resolve* into bucket i. If this happens, the histogram is updated (lines 2-5) by incrementing the count of the specific bucket i by $n_1 n_2$, where n_1, n_2 are the particle counts in cells M_1 and M_2, respectively. If the two cells do not resolve on the current density map, we move to a density map with higher (doubled) resolution and repeat the previous step. However, on this new density map, we try resolving all four partitions of M_1 with all those of M_2 (lines 12-16). In other words, there are $4 \times 4 = 16$ recursive calls to RESOLVETWOCELLS if M_1 and M_2 are not resolvable on the current density map. In another scenario, where M_1 and M_2 are not resolvable yet no more density maps are available, we have to calculate the distances of all particles in the nonresolvable cells (lines 6-11). The DM-SDH algorithm starts at the first density map DM_o whose cell diagonal length is smaller than the histogram bucket width p (line 2). It is easy to see that no pairs of cells are resolvable in density maps with resolution lower than that of DM_o. Within each cell on DM_o, we are sure that any intracell point-to-point distance is smaller than p; thus, all such distances are counted into the first bucket with range $[0, p)$ (lines 3-5). The algorithm proceeds by resolving intercell distances (i.e., calling RESOLVETWOCELLS) for all pairs of cells in DM_o (lines 6-7).

In DM-SDH, an important implementation detail that is relevant to our approximate algorithm design is the height of the quadtree (i.e., the number of density map levels). Recall that DM-SDH saves time by resolving cells such that we need not to calculate the point-to-point distances one by one. However, when the total point counts in a cell decreases, the time we save by resolving that cell also decreases. Imagine a cell with only four or fewer (eight for 3D data/space) data points; it does not give us any benefit in SDH query processing to further partition this cell on the next level: the cost of resolving the partitions could be higher than directly retrieving the particles and calculating distances (lines 7-11 in RESOLVETWOCELLS). Based on this observation, the total level of density maps H is set to be

$$H = \left\lceil \log_{2^d} \frac{N}{\beta} \right\rceil + 1, \tag{2}$$

where d is the number of dimensions, 2^d is the degree of the nodes in the tree (4/8 for 2D/3D data), and β is the average number of particles we desire in each leaf node. In practice, we set β to be slightly greater than 4 in 2D (8 for 3D data) because the CPU cost of resolving two cells is higher than computing the distance between two points.

3.2 Performance Analysis of DM-SDH

Clearly, by only considering atom counts in the density map cells (i.e., quadtree nodes), DM-SDH processes multiple point-to-point distances between two cells in one shot. This translates into significant performance improvement over the brute-force approach. We have accomplished a rigorous analysis of the performance of DM-SDH and derived its time complexity. The analysis focuses on the quantity of number of point-to-point distances that can be covered in resolved cells. We generate closed-form formulas for such quantities via a geometric modeling approach; therefore, rigorous analysis of the time complexity becomes possible. While the technical details of the analytical model are complex and can be found in a recent article [36], it is necessary to sketch the most important (and also most relevant) analytical results here for the purpose of laying out a foundation for the proposed approximate algorithm.

Theorem 1. *For any given standard SDH query with bucket width p, let DM_o be the first density map, where the DM-SDH algorithm starts running, and $\alpha(m)$ be the ratio of nonresolvable pairs of cells on a density map that lies m levels below DM_o (i.e., map DM_{o+m}) to the total number of cell pairs on that density map. We have*

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

Proof. See [36, Section 4]. □

What Theorem 1 tells us is: the chance that any pair of cells is not resolvable decreases by half with the density map level increases by one. In other words, for a pair of nonresolvable cells on DM_j where $j \geq o$, among the 16 pairs of subcells on the next level, we expect $16 \times \frac{1}{2} = 8$ pairs to be resolvable. Our analysis also shows that Theorem 1 not only works well for large l (i.e., smaller p, and more meaningful in simulation data analysis), but also *quickly converges even when l is reasonably small*. Furthermore, the above result is also true for *3D data* (see [36, Section 5.1]). The importance of Theorem 1 is in that it shows that the number of pairs of cells that do not resolve declines exponentially when the algorithm visits more levels of the density map. This is critical in studying the time complexity of DM-SDH that can be derived as follows: Given an SDH query with parameter p, the starting level DM_o is fixed. Suppose there are I nonresolvable pairs of cells on DM_o. On the next level DM_{o+1}, total number of cell pairs considered by the algorithm becomes $I2^{2d}$. According to Theorem 1, half of them will be resolved, leaving only $I2^{2d-1}$ pairs unresolved. On level DM_{o+2}, the number of nonresolvable pairs of cells becomes $\frac{I2^{2d-1}2^{2d}}{2} = I2^{2(2d-1)}$. Thus, after visiting the $n+1$ levels of the tree, the total number of calls to resolve cells made by DM-SDH is

$$\begin{aligned} T_c(N) &= I + I2^{2d-1} + I2^{2(2d-1)} + \cdots + I2^{n(2d-1)} \\ &= \frac{I[2^{(2d-1)(n+1)} - 1]}{2^{2d-1} - 1}. \end{aligned} \tag{3}$$

When N increases to $2^d N$, n increases by 1. Following the previous equation, we get

$$T_c(2^d N) = \frac{I[2^{(2d-1)(n+2)} - 1]}{2^{2d-1} - 1} = 2^{2d-1} T_c(N) - o(1),$$

which derives

$$T_c(N) = O\big(N^{\log_{2^d} 2^{2d-1}}\big) = O\big(N^{\frac{2d-1}{d}}\big).$$

The second part of the running time of DM-SDH involves the number of distances computed, which also follows the aforementioned recurrence relation. Details of such derivations can be found in [36, Section 6].

4 The Approximate DM-SDH Algorithm

In this section, we introduce a modified SDH algorithm to give such approximate results to gain better performance in return. Our solution targets at two must-have features of a decent approximate algorithm: 1) provable and controllable error bounds such that the users can have an idea on how close the results are to the fact; and 2) analysis of costs to

TABLE 2
Expected Percentage of Pairs of Cells that Can Be Resolved under Different Levels of Density Maps and Total Number of Histogram Buckets

Map levels	Total Number of Buckets				
	2	8	32	128	256
1	50.6565	52.5131	52.6167	52.6225	52.6227
2	74.8985	76.2390	76.3078	76.3112	76.3114
3	87.3542	88.1171	88.1539	88.1556	88.1557
4	93.6550	94.0582	94.0777	94.0778	94.0778
5	96.8222	97.0290	97.0285	97.0389	97.0389
6	98.4098	98.5145	98.5198	98.5195	98.5195
7	99.2046	99.2572	99.2596	99.2597	99.2597
8	99.6022	99.6286	99.6298	99.6299	99.6299
9	99.8011	99.8143	99.8149	99.8149	99.8149
10	99.9005	99.9072	99.9075	99.9075	99.9075

Computed with Mathematica 6.0.

reach (below) a given error bound, which guides desired performance/correctness tradeoffs.

In the DM-SDH algorithm, we have to: 1) keep resolving cells till we reach the lowest level of the tree; 2) calculate point-to-point distances when we cannot resolve two cells on the leaf level of the tree. Our idea for approximate SDH query processing is: *stop at a certain tree level and totally skip all distance calculations if we are sure that the number of distances in the unvisited cell pairs fall below some error tolerance threshold*. We name the new algorithm as ADM-SDH (that stands for Approximate Density Map-based SDH), and it can be easily implemented by modifying the DM-SDH algorithm. In particular, we stop the recursive calls to RESOLVETWOCELLS after m levels. The critical problem, however, is how to determine the value of m given a user-specified error tolerance bound ϵ. In this paper, we use the following metric to quantify the errors:

$$e = \frac{\sum_i |h_i - h_i'|}{\sum_i h_i},$$

where for any bucket i, h_i is the accurate count and h_i' the count given by the approximate algorithm. Obviously, we have $\sum_i h_i = \frac{N(N-1)}{2}$.

For any given density map DM_{o+m} and total number of buckets l, our analytical model (Theorem 1) gives the percentage of nonresolvable cell pairs $\alpha(m)$. Furthermore, due to the existence of a closed-form formula (see [36, Section 4.4]), $\alpha(m)$ can be efficiently computed. Table 2 lists some values of $1 - \alpha(m)$ (the percentage of *resolvable* cell pairs).

Given a user-specified error bound ϵ, we can find the appropriate levels of density maps to visit such that the unvisited cell pairs only contain less than $\epsilon\frac{N(N-1)}{2}$ distances. For example, for an SDH query with 128 buckets and error bound of $\epsilon = 3\%$, we get $m = 5$ by consulting the table. This means, to ensure the 3 percent error bound, we only need to visit five levels of the tree (excluding the starting level DM_o), and no distance calculation is needed. Table 2 serves as an excellent validation of Theorem 1: $\alpha(m)$ almost exactly halves itself when m increases by 1, even when l is as small as 2. Since the numbers on the first row (i.e., values for $1 - \alpha(1)$) are also close to 0.5, the correct choice of m for the guaranteed error rate ϵ is

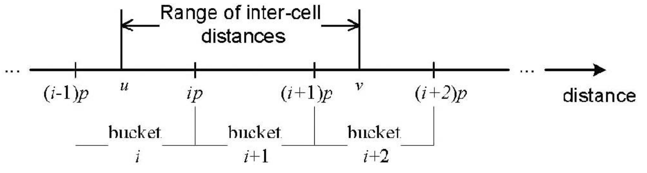


Fig. 2. Distance range of two resolvable cells overlap with three buckets.

$$m = \lg\frac{1}{\epsilon}. \tag{4}$$

The cost of the approximate algorithm only involves resolving cells on $m+1$ levels of the tree. Such cost can be derived following (3). After visiting $m+1$ levels of the tree, the total number of calls to resolve cells made by ADM-SDH is

$$T_c(N) = \frac{I\left[2^{(2d-1)(m+1)} - 1\right]}{2^{2d-1} - 1}. \tag{5}$$

Plugging (4) into (5), we have

$$T_c(N) \approx I2^{(2d-1)m} = I2^{(2d-1)\lg\frac{1}{\epsilon}} = I\left(\frac{1}{\epsilon}\right)^{2d-1}, \tag{6}$$

in which I is solely determined by the query parameter p. Therefore, we conclude that the running time of the ADM-SDH algorithm is not related to the input size N, but only to the user-defined error bound ϵ and the bucket width p.

4.1 Heuristic Distribution of Distance Counts

Now let us discuss how to deal with those nonresolvable cells after visiting $m+1$ levels on the tree. In giving the error bounds in our approximate algorithm, we are conservative in assuming that the distances in all the unresolved cells will be placed into the wrong bucket. In fact, this will almost never happen because we can distribute the distance counts in the unvisited cells to the histogram buckets heuristically and some of them will be done correctly. Consider two nonresolvable cells in a density map with particle counts n_1 and n_2 (i.e., total number of n_1n_2 distances between them), respectively. We know their minimum and maximum distances u and v (these are calculated beforehand in our attempt to resolve them) fall into multiple buckets. Fig. 2 shows an example that spans three buckets.

Using this example, we describe the following heuristics to distribute the n_1n_2 total distance counts into the relevant buckets. These heuristics are ordered in their expected correctness.

1. Put all n_1n_2 distance counts into one bucket that is predetermined (e.g., always putting the counts to the leftmost bucket); We name this heuristic as SKEW;
2. Evenly distribute the distance counts into the three buckets that $[u, v]$ overlaps, i.e., each bucket gets $\frac{1}{3}n_1n_2$; this heuristic is named EVEN;
3. Distribute the distance counts based on the overlaps between range $[u, v]$ and the buckets. In Fig. 2, the distances put into buckets i, $i+1$, and $i+2$ are $n_1n_2\frac{ip-u}{v-u}$, $n_1n_2\frac{p}{v-u}$, and $n_1n_2\frac{v-(i+1)p}{v-u}$, respectively. Apparently, by adapting this approach, we assume the (statistical) distribution of the point-to-point

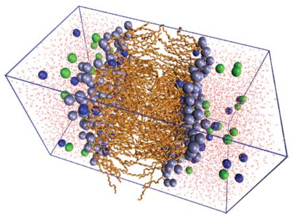


Fig. 3. The simulated hydrated dipalmitoylphosphatidylcholine bilayer system. We can see two layers of hydrophilic head groups (with higher atom density) connected to hydrophobic tails (lower atom density) are surrounded by water molecules (red dots).

distances between the two cells is uniform. This heuristic is called PROP (short for *proportional*).

The assumption of uniform distance distribution in PROP is obviously an oversimplification. In [19], we briefly mentioned a fourth heuristic: if we know the spatial distribution of particles within individual cells, we can generate the statistical distribution of the distances either analytically or via simulations, and put the n_1n_2 distances to involved buckets based on this distribution. This solution involves very nontrivial statistical inference of the particle spatial distribution and is beyond the scope of this paper.

Note that all above methods require only constant time to compute a solution for two cells. Therefore, the time complexity of ADM-SDH is not affected no matter which heuristic is used.

5 Empirical Evaluation of ADM-SDH

We have implemented the ADM-SDH algorithm using the C programming language and tested it with various synthetic/real data sets. The experiments are run at an Apple Mac Pro workstation with two dual-core 2.66-GHz Intel Xeon CPUs, and 8 GB of physical memory. The operating system is OS X 10.5 Leopard. In these experiments, we set the program to stop after visiting different levels of density maps and distribute the distances using the three heuristics (Section 4.1). We then compare the approximate histogram with those generated by regular DM-SDH. We use various synthetic and real data sets in our experiments. The synthetic data are generated from: 1) uniform distributions to simulate a system with particles evenly distributed in space; and 2) Zipf distribution with order 1 to introduce skewness to data spatial distribution.

The real data sets are extracted from a MS of biomembrane structures (Fig. 3). The data size in such experiments ranges from 50,000 to 12,800,000.

Table IV in Appendix I, which can be found on the Computer Society Digital Library at http://doi.ieeecomputersociety.org/10.1109/TKDE.2012.149, summarizes the range and default values of the parameters used in our experiments. The code of the algorithm and the data sets used in the experiments can be found in [39].

Fig. 5 shows the running time of ADM-SDH under one single p value of 2,500.0. Note that the "Exact" line shows the results of the basic DM-SDH algorithm, whose running time obviously increases polynomially with N at a slope of about 1.5. First, we can easily conclude that the running time after the tree construction stage does not change with the increase of data set size (Fig. 5a). The only exception is when m is 5—the running time increases when N is small and then stays as a constant afterwards. This is because the algorithm has less than five levels to visit in a bushy tree resulted from small N values. When N is large enough, running time no longer changes with the increase of N. In Fig. 5b, we plot the total running time that includes the time for quadtree construction. Under small m values, the tree construction time is a dominating factor because it increases with data size N (i.e., $O(N \log N)$). However, when $m > 3$, the shape of the curve does not change much as compared to those in Fig. 5a, indicating the time for running RESOLVETWOTREES dominates.

We observed surprising results on the accuracy of ADM-SDH. In Fig. 4, we plot the error rates observed in experiments with three different data sets and three heuristics mentioned in Section 4.1. First, it is obvious that less error was observed when m increases. The exciting fact is that, in almost all experiments, the error rate is lower than 10 percent—even for the cases of $m = 1$! These are much lower than the error bounds we get from Table 2. The correctness of heuristic SKEW is significantly lower than that of EVEN, and that of EVEN lower than PROP, as expected. Heuristic PROP achieves very low error rates even in scenarios with small m values. For all experiments, the increase of data size N does not cause an increase of the error rate of the algorithm. The above trends are observed in all three data sets. The interesting thing is, for the PROP experiments, we can even see the trend of decreasing error rates as N grows, especially for larger m values. We believe that serves as evidence of a (possibly) nice feature of the PROP heuristic. Our explanation is: when N is small, we could make a very big mistake in distributing the counts in individual operations. Imagine an extreme case in which 1 distance is to be distributed into two buckets—we could easily get a 100 percent error in the operation. Since the error compensation effects in PROP bring the overall error down to a very low level (as shown in Fig. 4), PROP is more sensitive to the errors of individual distribution operations than SKEW and EVEN are. More in-depth explorations of this phenomenon are worthwhile but also beyond the scope of this paper, in which we focus on an upper bound of the error.

5.1 Discussions

At this point, we can conclude that the ADM-SDH algorithm is an elegant solution to the SDH computation problem. According to our experiments, extremely low error rates can be obtained even when we only visit as few as one level of density map, leading to a very efficient algorithm yet with high accuracy in practice. It is clearly shown that the required running time for ADM-SDH grows very slowly with the data size N (i.e., only when m is of a small value does the tree construction time dominate).

The error rates achieved by ADM-SDH algorithm shown by current experiments are much lower than what we expected from our basic analysis. For example, Table 2 predicts an error rate of around 48 percent for the case of $m = 1$, yet the error we observed for $m = 1$ in our experiments is no more than 10 percent. With the PROP heuristic, this value can be as low as 0.5 percent. Our explanation for such low error rates is: in an individual operation to distribute the distance counts heuristically, we could have rendered a large error by putting too many

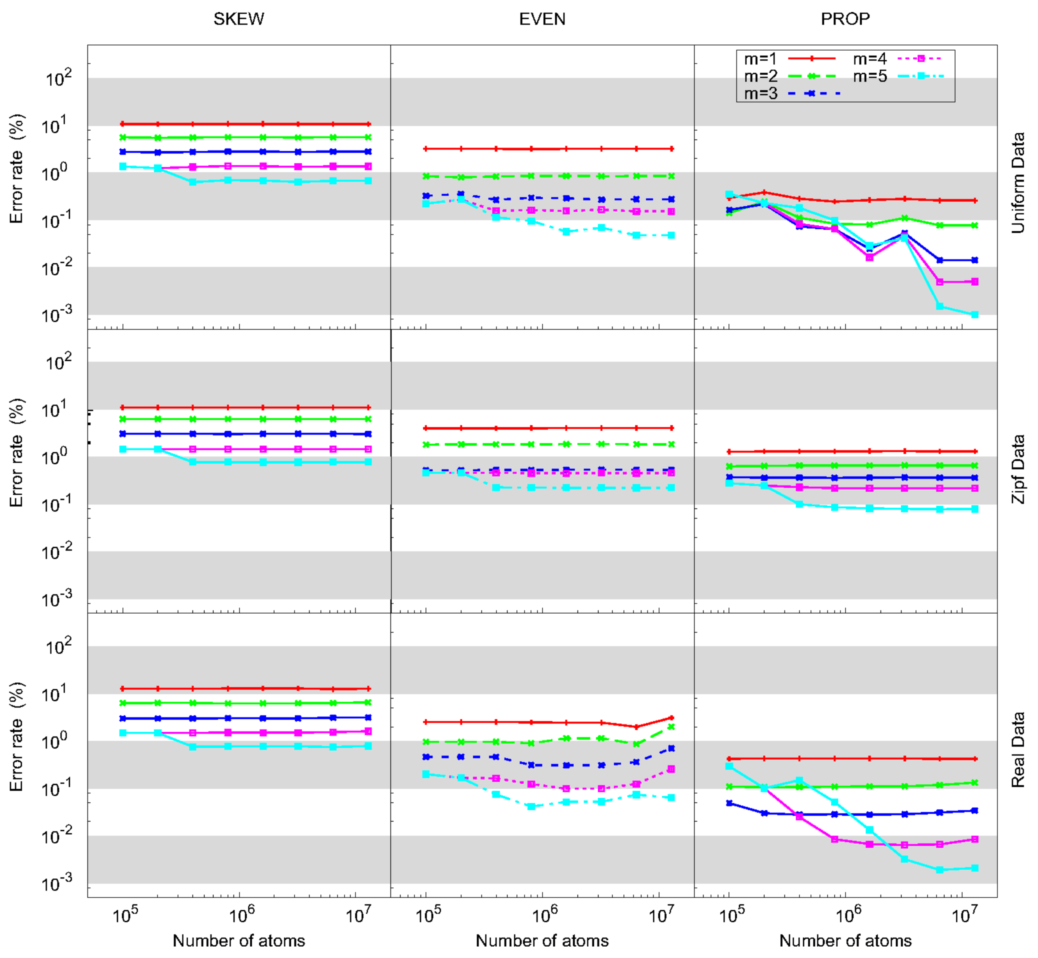


Fig. 4. Accuracy of ADM-SDH.

counts into a bucket (e.g., bucket i in Fig. 2) than needed. But the effects of this mistake could be (partially) canceled out by another distribution operation, in which too few counts are put into bucket i. Note that the total error in a bucket is calculated after all operations are processed; thus, it reflects the net effects of all positive and negative errors from individual operations. We call this phenomenon *error compensation*.

While more experiments under different scenarios are obviously needed, investigations from an analytical viewpoint are necessary. From the above facts, we understand that the bound given by Table 2 is loose. The real error bound should be described as

$$\epsilon = \epsilon'\epsilon'', \tag{7}$$

where ϵ' is the percentage of unresolved distances given by Table 2, and ϵ'' is the error rate created by the heuristics via error compensation. In the following section, we develop an analytical model to study how error compensation dramatically boosts accuracy of the algorithm.

6 Performance Analysis of ADM-SDH

It is difficult to obtain a tight error bound for ADM-SDH due to the fact that the error is related to data distribution. In this paper, we develop an analytical framework that achieves qualitative analysis of the behavior of ADM-SDH, with a focus on the generation of errors. Throughout the analysis, we assume uniform spatial distribution of particles and we consider only one level in the density map. At the start level (and the only level we visit), the side length of a cell is $\sqrt{2}p/2$.

6.1 The Distribution of Two Cells' Distance

We study two cells A and B on a density map, with cell A's row number denoted as t and column number as j,

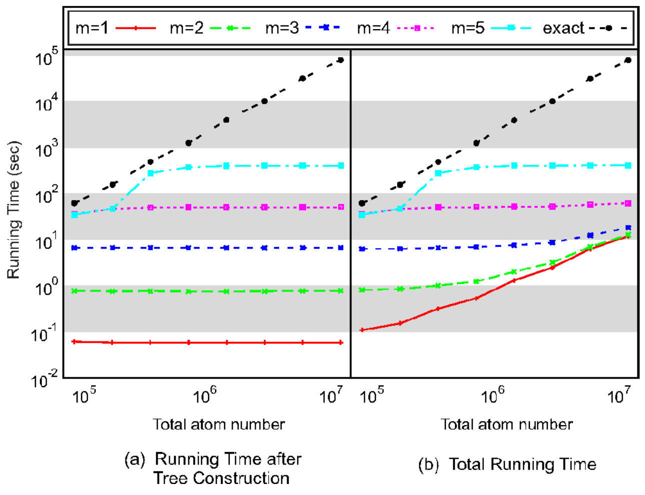


Fig. 5. Efficiency of ADM-SDH.

TABLE 3
The Buckets Involved in the Distribution of Distances from Two Nonresolvable Cells

Bucket Range	Cumulative Probabilities
$[\lfloor \frac{u}{p} \rfloor p, \lfloor \frac{u}{p} \rfloor p + p)$	$F(\lfloor \frac{u}{p} \rfloor p + p)$
$[\lfloor \frac{u}{p} \rfloor p + p, \lfloor \frac{u}{p} \rfloor p + 2p)$	$F(\lfloor \frac{u}{p} \rfloor p + 2p) - F(\lfloor \frac{u}{p} \rfloor p + p)$
$[\lfloor \frac{u}{p} \rfloor p + 2p, \lfloor \frac{u}{p} \rfloor p + 3p)$	$1 - F(\lfloor \frac{u}{p} \rfloor p + 2p)$

and cell B's row number as k and column number as l. We further denote the minimum distance between A and B as u, and the maximum distance as v. We propose the following lemma:

Lemma 1. *The range $[u, v]$ overlaps with at most three buckets in the SDH. In other words, $p <= v - u <= 2p$.*

The proof of Lema 1 can be found in Appendix II, available in the online supplemental material. By Lemma 1, we can easily see that v must fall into one of the two buckets with ranges $[\lfloor \frac{u}{p} \rfloor p + p, \lfloor \frac{u}{p} \rfloor p + 2p)$ and $[\lfloor \frac{u}{p} \rfloor p + 2p, \lfloor \frac{u}{p} \rfloor p + 3p)$.

Suppose the distance between points from the two cells follow a cumulative distribution function F over the range $[u, v]$, then the probabilities of a distance falling into the relevant bucket can be found in Table 3.

6.2 Compensating the Distance Counts in the SKEW Method

As mentioned earlier, an important mechanism that leads to low error rate in our algorithm is that the errors made by one distribution operation can be compensated by those of another. We can use the SKEW heuristic as an example to study this. In SKEW, all distance counts are put into one bucket, say, the one with the smallest bucket index. In other words, the distance counts in all three buckets (Table 3) are put into bucket with range $[\lfloor \frac{u}{p} \rfloor p, \lfloor \frac{u}{p} \rfloor p + p)$. The error would be large if we only consider this single distribution operation—by denoting the error as e, we have $e = 1 - F(\lfloor \frac{u}{p} \rfloor p + p)$ for the bucket $[\lfloor \frac{u}{p} \rfloor p, \lfloor \frac{u}{p} \rfloor p + p)$. The error e here is positive, meaning counts in the first bucket are overestimated. However, such errors can be canceled out by other distribution operations that move all distance counts from this bucket into another one. For example, if there exists another distribution operation with minimum distance $u_1 = u - p$, it would move some counts that belong to bucket 1 in Table 3 out, generating a negative error in bucket 1 and, thus, compensating the positive error mentioned before. Given this, an important goal of our analysis is to *find such compensating distribution operations and study how much error can be compensated*. We first show that under an ideal situation the error can reach zero.

Lemma 2. *For any distribution operation with minimum distance u, if there exists another such operation with minimum distance $u_1 = u - p$, the error generated by ADM-SDH using the SKEW approach is zero.*

Proof. According to Table 3, for any distribution operation, the error to the first SDH bucket (denoted as bucket i) it involves is $1 - F(\lfloor \frac{u}{p} \rfloor p + p)$, and this error is positive (i.e., overestimation). Suppose that there is another distribution with minimum distance $u_1 = u - p$, then this operation generates a negative error $F(\lfloor \frac{u}{p} \rfloor p + 2p) - F(\lfloor \frac{u}{p} \rfloor p + p)$ to bucket i. For the same reason, a third distribution with minimum distance $u_2 = u - 2p$ generates a negative error of $1 - F(\lfloor \frac{u}{p} \rfloor p + 2p)$. It is easy to see that the combined error (by putting all relevant negative and positive errors together) to bucket i is 0. □

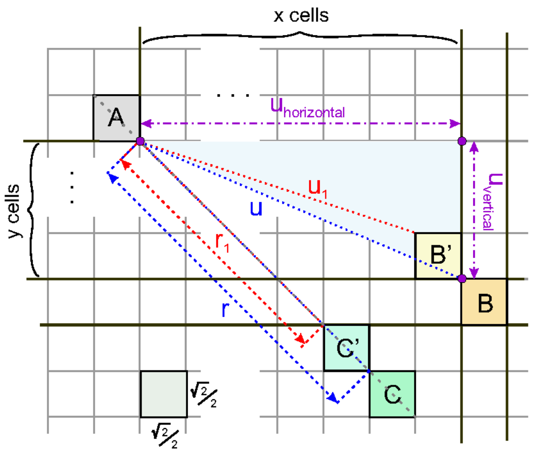


Fig. 6. Pairs of cells that lead to total or partial error compensation.

An example of two cells that contribute to each other's error compensation in the aforementioned way can be seen in Fig. 6. Namely, the cells C and C' when we compute the minimum distances AC and AC'.

Unfortunately, the above condition of the existence of a u_1 value that equals $u - p$ cannot be satisfied for all pairs of cells. From Lemma 2, however, we can easily see that the error is strongly related to the quantity u. In the following text, we study how the errors can be partially compensated by neighboring pairs of cells.

Without loss of generality, we take any pair of cells in the density map that are x cells apart horizontally and y cells apart vertically, such as cells A and B in Fig. 6.

For the convenience of presentation, we define p to be a unit ($p = 1$ *unit*). Given that fact, a cell's side is $\frac{\sqrt{2}}{2}$. Following this, the horizontal and vertical distances between A and B are $u_{horizontal} = \frac{\sqrt{2}}{2}x$ and $u_{vertical} = \frac{\sqrt{2}}{2}y$, respectively, as shown in Fig. 6. Thus, the minimum distance between the above two cells can be written as

$$u = \sqrt{u_{horizontal}^2 + u_{vertical}^2} = \sqrt{\frac{x^2}{2} + \frac{y^2}{2}}. \tag{8}$$

The critical observation that leads to the success of our analysis is obtained by studying another cell, such as cell B' in Fig. 6. Its minimum distance to the cell A is

$$u_1 = \sqrt{\frac{(x-1)^2}{2} + \frac{(y-1)^2}{2}}.$$

Let us denote the quantity $u - u_1$ as Δ. We have

$$\begin{aligned}\Delta = u - u_1 &= \frac{(u - u_1)(u + u_1)}{u + u_1} = \frac{u^2 - u_1^2}{u + u_1} \\ &= \frac{\frac{x^2}{2} + \frac{y^2}{2} - \frac{(x-1)^2}{2} - \frac{(y-1)^2}{2}}{u + u_1} \approx \frac{x + y - 1}{2u}.\end{aligned} \tag{9}$$

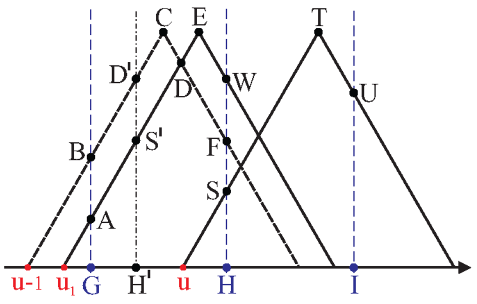


Fig. 7. Distance compensation between two pairs of cells.

Suppose $x >= y$ and $z = y/x$, (9) can be rewritten as

$$\begin{aligned}\Delta &\approx \frac{x+y-1}{2u} = \frac{\frac{x+y-1}{x}}{\frac{2\sqrt{\frac{x^2}{2}+\frac{y^2}{2}}}{x}} = \frac{1+\frac{y}{x}-\frac{1}{x}}{\sqrt{4\left(\frac{x^2}{2x^2}+\frac{y^2}{2x^2}\right)}} \\ &= \frac{1+z-\frac{1}{x}}{\sqrt{2+2z^2}} \approx \frac{1+z}{\sqrt{2+2z^2}}.\end{aligned} \tag{10}$$

Although x and y are integers, we can treat z as a continuous variable due to the existence of a large number of possible values of x and y in a density map with many cells. Since $d\Delta/dz > 0$, we conclude that Δ increases with z. Two boundary cases are: 1) when $y = 0$ we have $z = 0$ and $\Delta = \frac{\sqrt{2}}{2}$; 2) when $y = x$, we have $z = 1$ and $\Delta = 1$.

According to Lemma 2, the error of the SKEW method, e_{skew}, can be approximately noted by the difference between one and Δ, $e_{skew} \approx 1 - \Delta$. This error, e_{skewed}, ranges from 0 to $1 - \frac{\sqrt{2}}{2}$, since Δ ranges from $\frac{\sqrt{2}}{2}$ to 1. The compensating process in distance counts is shown in Fig. 6. As mentioned before, C and C' are examples of two cells for which the difference of minimum distances to cell A is 1 and the cells B and B' are two cells for which the difference of minimum distances is different from one (less than one in this case).

Let us analyze the minimum distances u and u_1 from cell pairs (A, B) and (A, B'), respectively. We know that each such pair of minimum distances (u, u_1) with property $u - u_1 \neq 1$ generates an error. In the next few paragraphs, we will quantitatively approximate the error generated by such pair of minimum distances, and also show that the sum of all such errors is small (can be qualitatively bounded).

Since we want to give an analytical description of a quantity ($1 - \Delta$ *or* $1 - (u - u_1)$), we need to use the underlying distribution of that quantity. The distribution of the distances between points from cell A and cell B (or B') can be viewed as noncentral chi-squared. Without loss of generality, we can choose one point from cell A and make it a base point with coordinates $(0, 0)$. The distribution of the distances between this base point and points from cell B (or B') can be regarded as triangular.

We use Fig. 7 to geometrically describe the background of the error compensation in our method that leads to lower error. The triangles in Fig. 7 represent the triangular density distributions of the distances between the base point and the points of three cells. The bases of the three triangles represent the $[min, max]$ ranges that the distances between points from a particular cell and the base point fall in. In our analysis, we define G as $\lfloor \mathrm{u} \rfloor$, and we also define $H = G + 1$ and $I = H + 1$. The triangles STU, AEW, and BCF are the density distributions of u, u_1, and $u - 1$, respectively. The line $H'S'D'$ is the symmetry line of WFH with respect to the vertical line which passes through the point D.

As we know, if $u_1 = u - 1$, the error of distance counts produced by the period between H and I can be compensated by the one produced by the period between G and H. In other words, according to SKEW, when we count the number of distances between two cells with minimum distance u, the number of distances between H and I is added and leads to a positive error. When we count the number of distances between two cells with minimum distance $u - 1$, the number of distances between G and H is missed and leads to a negative error. If the distribution is the same, the area of $GBCFH$ is the same as that of $HSTUI$, and no error would be produced.

If $u_1 \neq u - 1$, there is difference between the area of $GAEWH$ and the area of $HSTUI$. In other words, there is difference between the area of $GAEWH$ and the area of $GBCFH$. This difference can be computed by the area of $ABD'S'$ and that represents the error imposed by the difference between u_1 and $u - 1$, which we denote as $e_{u_1,u-1}$. We know that $CE = u_1 - u + 1$ and $GH' < u - u_1 = \Delta$. Considering the fact that the area of each of the three triangles in Fig. 7 is 1, the height of each of these triangles is $\frac{2}{v-u}$ (because the base is $v - u$). Therefore, the ratio $\frac{AB}{CE}$ has the following value (more details in Appendix III, available in the online supplemental material):

$$\frac{AB}{CE} = \frac{\frac{2}{v-u}}{\frac{v-u}{2}} = \frac{4}{(v-u)^2}. \tag{11}$$

Furthermore, the length of AB is

$$AB = \frac{4}{(v-u)^2} * CE = \frac{4(u_1 - u + 1)}{(v-u)^2} = \frac{4(1-\Delta)}{(v-u)^2}. \tag{12}$$

The area of $ABD'S'$, thus the error $e_{u_1,u-1}$, can be computed as follows:

$$e_{u_1,u-1} = AB * GH' < \frac{4(1-\Delta) * \Delta}{(v-u)^2} \tag{13}$$

$$e_{u_1,u-1} < \frac{(1-\Delta)}{\Delta} = \frac{1}{\Delta} - 1. \tag{14}$$

Therefore, the accumulated error, $e_{\overline{z=0,1}}$ over the range of z ($z \in [0, 1]$) can be computed by the following equation (considering (10) for Δ):

$$\begin{aligned}e_{\overline{z=0,1}} &= \sum_{z=0}^{1} e_{u_1,u-1} = \sum_{z=0}^{1}\left(\frac{1}{\Delta} - 1\right) \\ &\approx \sum_{z=0}^{1}\left(\frac{\sqrt{2+2z^2}}{1+z} - 1\right).\end{aligned} \tag{15}$$

When $0 \leq z \leq 1$, we can use well-established mathematical tools (Fig. 8) to approximate $e_{\overline{z=0,1}}$ as follows:

$$e_{\overline{z=0,1}} \leq \sum_{z=0}^{1} 0.211 * (1-z)^5 + 0.211 * (1-z)^2, \tag{16}$$

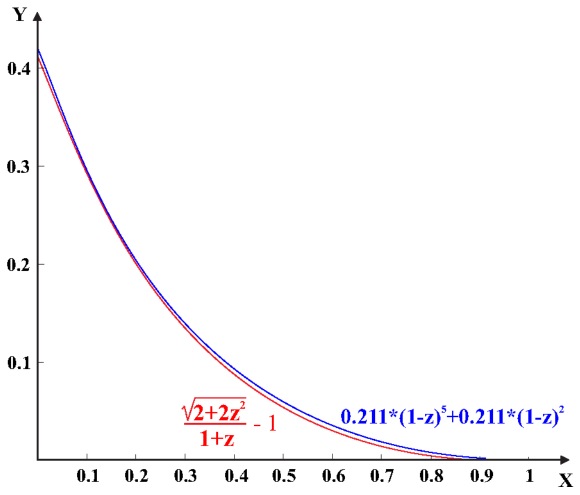


Fig. 8. Approximation of $\frac{1}{\Delta} - 1$ obtained in Matlab.

treating z as continuous variable we get

$$e_{z=\overline{0,1}} \leq \int_0^1 (0.211 * (1-z)^5 + 0.211 * (1-z)^2)\, dz \\ = 0.1055. \tag{17}$$

Equation (17) means that the total error rendered by the SKEW under the assumptions we stated in the beginning of Section 6 is less than 10.55 percent. Due to the assumptions we made, we do not claim this as a rigorous bound. However, it clearly shows that our algorithm is able to produce really good results with low errors by visiting only one level in the density map. And this conclusion builds the foundation of an improved approximate algorithm (Section 7).

One special note here is that (17) does not cover the cases in which the minimum distance u falls into the first SDH bucket (i.e., $u < p$). However, our analysis shows that such cases do not impact the results in (17) significantly. More details can be found in Appendix IV, available in the online supplemental material.

7 Single-Level Approximate Algorithm

Via the performance analysis of ADM-SDH, and looking back to the error bound described in (7), we concluded that ϵ'' is very small. Even if we allow ϵ' to be 100 percent, meaning no cell resolution is possible, we can still achieve low and controllable error rates in our results. Based on such conclusion, we introduce an improved approximate algorithm we call *single-level SDH algorithm* (SL-SDH). There are two major differences between the two algorithms: 1) the number of levels (density maps) each algorithm visits: unlike ADM-SDH, which visits $m+1$ levels, SL-SDH visits only one level of the tree, *and is thus given the name SL-SDH*. The single level visited by the SL-SDH is a user-defined variable and can be any level of the tree. 2) the starting level of the algorithms: ADM-SDH starts at a predetermined level, based on the bucket width p and the maximum distance between any two points of the system. SL-SDH, on the other hand, starts at a user-defined level (and visits only that level), which can be any level of the tree. SL-SDH improves over ADM-SDH in two important aspects. First, we only need a single DM that can be built in $O(N)$ time (instead of the $O(N \log N)$ time needed to build the quadtree). Second, we reduce the posttree-construction running time of the algorithm, with little increase of the error, as we only run RESOLVETWOTREES for cells in one density map (i.e., a single level).

A special note here is that the running time of SL-SDH is no longer determined by the bucket width p. Recall that ADM-SDH starts at the density map DM_o, where the diagonal of a single cell is less than or equal to p. When p is small, the number of cells in DM_o is large, and we have to invoke the RESOLVETWOCELLS procedure more times. To remedy this, we allow SL-SDH to run on a (single) density map above DM_o, i.e., one with larger cell sizes (and fewer cells). This is based on a hypothesis motivated by our performance analysis of ADM-SDH: the error compensation mechanism we studied will also work for density maps above DM_o. We know the error is very small for running RESOLVETWOTREES for those cells in DM_o—doing the same on higher level density maps should still render reasonable (although higher) error rates. Unfortunately, an analytical study of such errors is very difficult. In the remainder of this section, we empirically evaluate the error and time tradeoff of the final version of the SL-SDH algorithm.

7.1 Experimental Results

We have implemented the SL-SDH algorithm using the C programming language and tested it with various synthetic/real data sets. The experiments are run in the same environment as the experiments for the ADM-SDH in Section 5.

Since we know, from our previous experiments, that the PROP heuristic for distributing the distances in nonresolvable cells produces the best results, we have only used that heuristic to show the results of the single level approximate algorithm. We have run the algorithm on two synthetic data sets (with uniform and skewed distribution of atoms) under five different N values (i.e., 1, 3, 5, 7.5, and 12 million). We also ran the algorithm on one real simulation data set with 891,272 atoms.

Fig. 9 shows the results from the experiments on uniform and skewed data, respectively, with 7.5 million atoms and the real data with 891,272 atoms. From this figure, we can see that the error rate decreases when we increase the level in the density map. We also see that the error rate decreases when the bucket width increases.

Fig. 10 demonstrates the effects of system size N on the accuracy of SL-SDH. Each line in Fig. 10 plots the error rates of SL-SDH when run at a particular level of density map under a particular atom count. We can easily see that the lines for the five different system sizes (of the same density map) are very similar, giving rise to one cluster of lines for each level of density map. Fig. 10 shows that the error introduced by the SL-SDH algorithm is not affected by the number of atoms in the data set N, but by the level of density map the algorithm works at. On the other hand, the running time of the SL-SDH algorithm was also found to be independent of N, as shown in Fig. 11. The only exceptions are for levels 3 and 4, in which the tree construction time dominates.

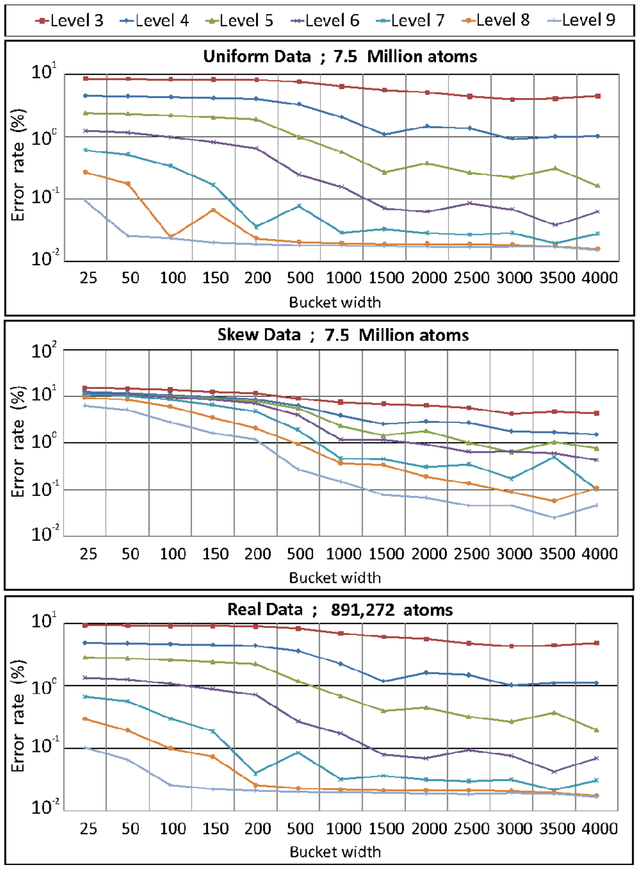


Fig. 9. Accuracy of SL-SDH under different bucket width for synthetic (uniform and skewed) and real data.

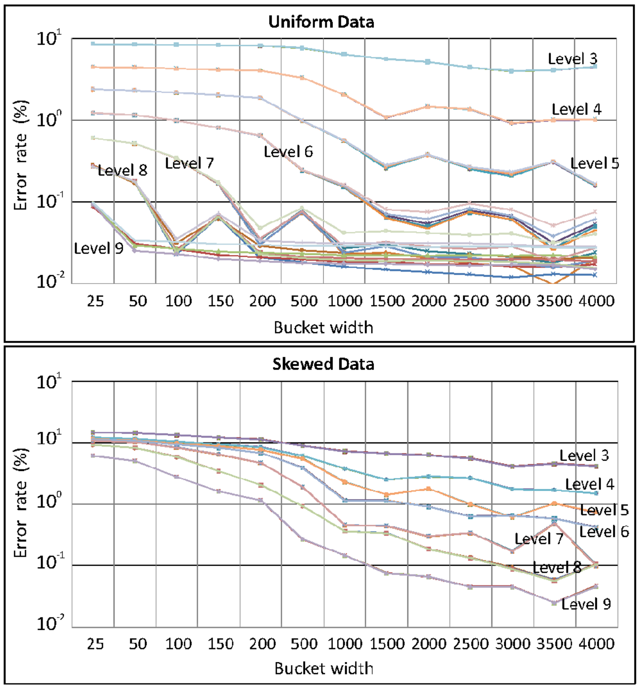


Fig. 10. Accuracy of SL-SDH under different bucket width and different atom counts for synthetic data (uniform and skewed).

The difference between SL-SDH and ADM-SDH can be seen by comparing Fig. 11 with Fig. 5, and Fig. 10 with Fig. 4. However, to better understand the accuracy/performance tradeoffs provided by the two algorithms, we introduce a new metric we call *Error Delay Product (EDP)*, which is defined as the product of the error rate and the running time of the algorithm. Obviously, higher EDP means worse accuracy/performance tradeoff. Fig. 12 shows the EDPs for both algorithms ran under four different bucket widths (i.e., 100, 500, 1,000, and 2,000). It is obvious that the SL-SDH algorithm produces better EDP than the regular approximate algorithm. This is especially the case when the bucket width is small—the EDPs of SL-SDH are orders of magnitude lower than those provided by ADM-SDH. The reason for this is that the ADM-SDH algorithm starts at a level predetermined by the bucket width p, whereas SL-SDH can start at any level the user wants (this is why the number of levels shown in Fig. 12 changes for ADM-SDH and stays constant for SL-SDH for different bucket widths). In other words, when p gets smaller, ADM-SDH has to start at a lower level of the tree (higher in number) with more cells to process which makes the running time longer, thus increasing the EDP. As we can see from Fig. 12, when the bucket width is 100, ADM-SDH can only work in level 9 (its starting level DM_o). This yields long running time. On the other hand, the SL-SDH can work at any user-defined level of the tree, one that yields the desired accuracy while keeping the running time low. The other three parts of Fig. 12 show that as the bucket width grows, the EDPs of the two algorithms get closer. Nevertheless, there is always an EDP of the SL-SDH that is better than the EDP of the ADM-SDH. In general, the EDPs of both algorithms decrease with the decrease of the density map level (level 9 always showing the worst tradeoff). However, the benefit of the SL-SDH is that it can work at any user-defined level of the tree, and our experiments show that its accuracy is already high when working on a coarse density map.

In summary, our experimental results convey three important messages. First, the SL-SDH algorithm significantly improves the accuracy/performance tradeoff over ADM-SDH. Such improvements are more obvious under small SDH bucket width. This is very important: the computation of SDHs is generally preferred to be done under smaller p values as it carries more information about

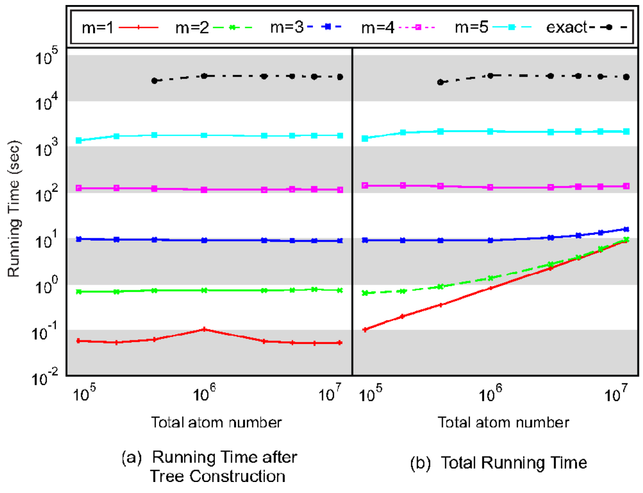


Fig. 11. Efficiency of SL-SDH.

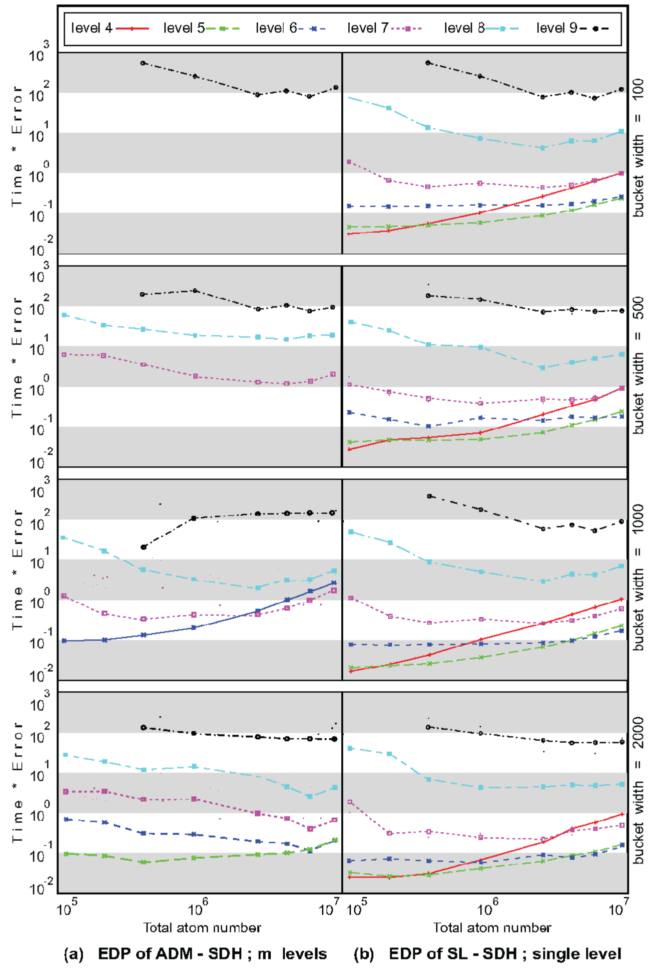


Fig. 12. Accuracy and performance tradeoffs of ADM-SDH and SL-SDH algorithms under different SDH bucket widths.

the distribution of the distances. Second, users can choose the appropriate (single) level among all the density maps to run the algorithm based only on the desired accuracy. Third, we also show that, like ADM-SDH, the running time and the error rate of SL-SDH are not affected by the number of atoms in the data set.

8 Conclusions and Future Work

The main objective of our work is to accomplish efficient computation of SDH, a popular quantity in particle simulations, with guaranteed accuracy. In this paper, we introduce approximate algorithm for SDH query processing based on our previous work developed around a Quadtree-like data structure named *density map*. The experimental results show that our approximate algorithm has very high performance (short running time) while delivering results with astonishingly low error rates. Aside from the experimental results, we also analytically evaluate the performance/accuracy tradeoffs of the algorithm. Such analyses showed that the running time of our algorithm is completely independent of the input size N, and derived a provable error bound under desired running time. We further developed another mathematical model to perform in-depth study of the mechanism that leads to low error rates of the algorithm. Aside from administering tighter bounds (under some assumptions) on the error of the basic approximate algorithm, our model also gives insights on how the basic algorithm can be improved. Following these insights, a new single-level approximate algorithm with improved time/accuracy tradeoff was proposed. Our experimental results supported our analysis. Having these experimental results on hand, one aspect of our future work will be to establish a provable error bound for the new algorithm.

Many times, the MS systems are observed over certain period of time and SDH computation is required for every frame (time instance) over that period. Therefore, another direction of our on-going work is to efficiently compute the SDHs of consecutive frames by taking advantage of the temporal locality of data points. We can also extend our work to the computation of m-body correlation functions with $m > 2$—a more general form of spatial statistics that involves counting all possible m-particle tuples.

Acknowledgments

The project described was supported by an Award Number R01GM086707 from the National Institute Of General Medical Sciences (NIGMS) at the National Institutes of Health (NIH). The authors would like to thank Anand Kumar who has contributed his time and knowledge toward realization of this paper. These authors contributed equally to this work.

References

[1] J. Gray, D. Liu, M. Nieto-Santisteban, A. Szalay, D. DeWitt, and G. Heber, "Scientific Data Management in the Coming Decade," *ACM SIGMOD Record*, vol. 34, no. 4, pp. 34-41, Dec. 2005.
[2] A.S. Szalay, J. Gray, A. Thakar, P.Z. Kunszt, T. Malik, J. Raddick, C. Stoughton, and J. vandenBerg, "The SDSS Skyserver: Public Access to the Sloan Digital Sky Server Data," *Proc. ACM SIGMOD Int'l Conf. Management of Data*, pp. 570-581, 2002.
[3] M.Y. Eltabakh, M. Ouzzani, and W.G. Aref, "BDBMS - A Database Management System for Biological Data," *Proc. Third Biennial Conf. Innovative Data Systems Resarch (CIDR)*, pp. 196-206, 2007.
[4] M.H. Ng, S. Johnston, B. Wu, S.E. Murdock, K. Tai, H. Fangohr, S.J. Cox, J.W. Essex, M.S.P. Sansom, and P. Jeffreys, "BioSimGrid: Grid-Enabled Biomolecular Simulation Data Storage and Analysis," *Future Generation Computer Systems*, vol. 22, no. 6, pp. 657-664, June 2006.
[5] J.M. Patel, "The Role of Declarative Querying in Bioinformatics," *OMICS: A J. Integrative Biology*, vol. 7, no. 1, pp. 89-91, 2003.
[6] S. Klasky, B. Ludaescher, and M. Parashar, "The Center for Plasma Edge Simulation Workflow Requirements," *Proc. IEEE Workshop Workflow and Data Flow for Scientific Applications (SciFlow '06)*, pp. 73-73, 1991.
[7] J.L. Stark and F. Murtagh, *Astronomical Image and Data Analysis.* Springer, 2002.
[8] M.P. Allen and D.J. Tildesley, *Computer Simulations of Liquids.* Clarendon Press, 1987.
[9] D. Frenkel and B. Smit, *Understanding Molecular Simulation: From Algorithm to Applications,* series Computational Science Series, vol. 1. Academic Press, 2002.
[10] J.M. Haile, *Molecular Dynamics Simulation: Elementary Methods.* Wiley 1992.
[11] D.P. Landau and K. Binder, *A Guide to Monte Carlo Simulation in Statistical Physics.* Cambridge Univ. Press, 2000.
[12] P.K. Agarwal, L. Arge, and J. Erikson, "Indexing Moving Objects," *Proc. Int'l Conf. Principles of Database Systems (PODS)*, pp. 175-186, 2000.
[13] M. Bamdad, S. Alavi, B. Najafi, and E. Keshavarzi, "A New Expression for Radial Distribution Function and Infinite Shear Modulus of Lennard-Jones Fluids," *Chemical Physics*, vol. 325, pp. 554-562, 2006.

[14] J.P. Hansen and I.R. McDonald, *Theory of Simple Liquids.* Academic Press, 2006.

[15] A. Filipponi, "The Radial Distribution Function Probed by X-Ray Absorption Spectroscopy," *J. Physics: Condensed Matter,* vol. 6, pp. 8415-8427, 1994.

[16] B. Hess, C. Kutzner, D. van der Spoel, and E. Lindahl, "GROMACS 4: Algorithms for Highly Efficient, Load-Balanced, and Scalable Molecular Simulation," *J. Chemical Theory and Computation,* vol. 4, no. 3, pp. 435-447, Mar. 2008.

[17] V. Springel, S.D.M. White, A. Jenkins, C.S. Frenk, N. Yoshida, L. Gao, J. Navarro, R. Thacker, D. Croton, J. Helly, J.A. Peacock, S. Cole, P. Thomas, H. Couchman, A. Evrard, J. Colberg, and F. Pearce, "Simulations of the Formation, Evolution and Clustering of Galaxies and Quasars," *Nature,* vol. 435, pp. 629-636, June 2005.

[18] A.G. Gray and A.W. Moore, "N-Body Problems in Statistical Learning," *Proc. Advances in Neural Information Processing Systems (NIPS),* pp. 521-527, 2000.

[19] Y.-C. Tu, S. Chen, and S. Pandit, "Computing Distance Histograms Efficiently in Scientific Databases," *Proc. IEEE 25th Int'l Conf. Data Eng. (ICDE),* pp. 796-807, Mar. 2009.

[20] M. Arya, W.F. Cody, C. Faloutsos, J. Richardson, and A. Toya, "QBISM: Extending a DBMS to Support 3D Medical Images," *Proc. 10th Int'l Conf. Data Eng. (ICDE),* pp. 314-325, 1994.

[21] M. Stonebraker, S. Madden, D.J. Abadi, S. Harizopoulos, N. Hachem, and P. Helland, "The End of an Architectural Era (It's Time for a Complete Rewrite)," *Proc. 33rd Int'l Conf. Very Large Data Bases (VLDB),* pp. 1150-1160, 2007.

[22] B. Howe, D. Maier, and L. Bright, "Smoothing the ROI Curve for Scientific Data Management Applications," *Proc. Third Biennial Conf. Innovative Data Systems Research (CIDR),* pp. 185-195, 2007.

[23] P.G. Brown, "Overview of SciDB: Large Scale Array Storage, Processing and Analysis," *Proc. ACM SIGMOD Int'l Conf. Management of Data,* pp. 963-968, 2010.

[24] M. Feig, M. Abdullah, L. Johnsson, and B.M. Pettitt, "Large Scale Distributed Data Repository: Design of a Molecular Dynamics Trajectory Database," *Future Generation Computer Systems,* vol. 16, no. 1, pp. 101-110, Jan. 1999.

[25] J. Barnes and P. Hut, "A Hierarchical O(N log N) Force-Calculation Algorithm," *Nature,* vol. 324, no. 4, pp. 446-449, 1986.

[26] L. Greengard and V. Rokhlin, "A Fast Algorithm for Particle Simulations," *J. Computational Physics,* vol. 135, no. 12, pp. 280-292, 1987.

[27] P.B. Callahan and S.R. Kosaraju, "A Decomposition of Multi-Dimensional Point Sets with Applications to K-Nearest-Neighbors and N-Body Potential Fields," *J. ACM,* vol. 42, no. 1, pp. 67-90, 1995.

[28] L. Golab and T. Özsu, *Data Stream Management - Synthesis Lectures on Data Management.* Morgan and Claypool, 2010.

[29] A.L.-O. Demaine and J.I. Munro, "Frequency Estimation of Internet Packet Streams with Limited Space," *Proc. 10th Ann. European Symp. Algorithms,* pp. 348-360, 2002.

[30] G.S. Manku and R. Motwani, "Approximate Frequency Counts over Data Streams," *Proc. 28th Int'l Conf. Very Large Data Bases,* pp. 346-357, 2002.

[31] S.R. Manku and B. Lindsay, "Random Sampling Techniques for Space Efficient Online Computation of Order Statistics of Large Data Sets," *Proc. ACM SIGMOD Int'l Conf. Management of Data,* pp. 251-262, 1999.

[32] P.B. Gibbons, "Distinct Sampling for Highly-Accurate Answers to Distinct Values Queries and Event Reports," *Proc. 27th Int'l Conf. Very Large Data Bases,* pp. 541-550, 2001.

[33] A. Pavan and S. Tirthapura, "Range-Efficient Computation of F0 over Massive Data Streams," *Proc. 21st Int'l Conf. Data Eng.,* pp. 32-43, 2005.

[34] P. Flajolet and G.N. Martin, "Probabilistic Counting," *Proc. IEEE Conf. Foundations of Computer Science,* pp. 76-82, 1983.

[35] J. Zhou, J. Sander, Z. Cai, L. Wang, and G. Lin, "Finding the Nearest Neighbors in Biological Databases Using Less Distance Computations," vol. 7, no. 4, pp. 669-680, 2010.

[36] S. Chen, Y.-C. Tu, and Y. Xia, "Performance Analysis of a Dual-Tree Algorithm for Computing Spatial Distance Histograms," *The VLDB J.,* vol. 20, no. 4, pp. 471-494, 2011.

[37] A. Kumar, V. Grupcev, Y. Yuan, Y.-C. Tu, and G. Shen, "Distance Histogram Computation Based on Spatiotemporal Uniformity in Scientific Data," *Proc. 15th Int'l Conf. Extending Database Technology,* pp. 288-299, 2012.

[38] J.A. Orenstein, "Multidimensional Tries Used for Associative Searching," *Information Processing Letters,* vol. 14, no. 4, pp. 150-157, 1982.

[39] Y.-C. Tu, V. Grupcev, and A. Kumar, "Algorithm's Code and Data Sets," www.cse.usf.edu/vgrupcev/tkde2012.zip, 2011.

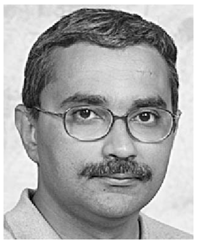
Vladimir Grupcev received the bachelor's degree in applied mathematics and computer science from the University of Ss. Cyril and Methodius, Macedonia, in 2005, the master's degree in mathematics from the University of South Florida in 2007, and is currently working toward the PhD degree in the Department of Computer Science and Engineering at the University of South Florida. His area of interest includes scientific data management and high-performance computing. He is a student member of the IEEE.

Yongke Yuan received the PhD degree in 2003 in management engineering from Peking University in China, performed the postdoctoral studies at Beijing University of Technology, and was a visiting scholar at the University of South Florida. He is an associate professor in the Department of Economics and Management at Beijing University of Technology in Beijing, China. Most of his research has focused on industry economics such as CGE Model for Chinese industries. His current focus is coupling natural and social system science with engineering to forecast the development of Chinese industries.

Yi-Cheng Tu received the bachelor's degree in horticulture from Beijing Agricultural University, China, and the MS and PhD degrees in computer science from Purdue University in 2003 and 2007, respectively. He is currently an assistant professor in the Department of Computer Science and Engineering at the University of South Florida, Tampa, Florida. His current research addresses energy-efficient database systems, scientific data management, and high-performance computing. He also worked on data stream management systems, self-tuning databases, peer-to-peer systems, and multimedia databases. He is a member of the IEEE, the ACM/SIGMOD, and the ASEE.

Jin Huang received the BS degree in mathematics from Dalian University of Technology, China, in 2004, and is currently working toward the PhD degree in computer science at the University of Texas at Arlington. His research interests include machine learning, data mining, and medical informatics. He is a student member of the IEEE.

Shaoping Chen received the bachelor's degree in mathematics and the PhD degree in ship and marine structures design from Wuhan University of Technology, China, in 1982 and 2002, respectively. He is currently an associate professor in the Department of Mathematics at Wuhan University of Technology, Wuhan, China. He conducts research in applied mathematics, computer-aided geometric design, and high-performance computing.



Sagar Pandit received the PhD degree from the Department of Physics, University of Pune, in 1999 in the area of dynamical systems. He is an assistant professor of physics at the University of South Florida. Later, his research shifted to biological physics, specifically membrane physics. His current research interests include biomembranes, mathematical modeling of complex biological and social systems, dynamical systems, and computational approaches toward addressing problems in these fields.

Michael Weng received the BS degree in engineering science from the Shandong University of Science and Technology (SUST), Shandong, China, in 1982, the MS degree from SUST in 1984, and the PhD degree from the Pennsylvania State University in 1994. He has been a member of the Department of Industrial and Management Systems Engineering Faculty at the University of South Florida (USF) since 1995. Prior to joining the faculty at USF, he was a senior manufacturing engineer at Ford Motor Company. His primary areas of research and scholarly work are in the fields of production planning and scheduling, supply chain management, optimization, discrete event simulation, and statistical modeling.