Computing Distance Histograms Efficiently in Scientific Databases

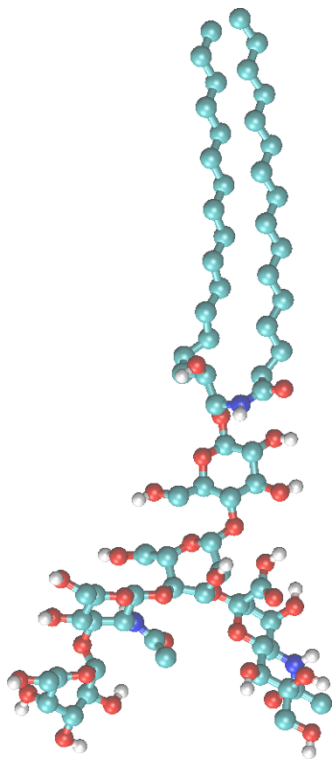
Yicheng Tu,* Shaoping Chen, *§ and Sagar Pandit*

* University of South Florida, Tampa, FL, U.S.A.

§ Wuhan University of Technology, Wuhan, Hubei, China

Molecular Simulations (MS)

- Large scale biological structures are represented using all the individual atoms.
- Data is stored in single or multiple trajectory databases containing time frames.
- Each frame is a sequential list of atoms with their positions, velocities, perhaps forces, masses, and types.
- Dataset is very large: millions of atoms, tens of thousands of frames.
- Similar methodology in other sciences: astronomy, material science, civil engineering



Querying a MS Database

- Mainstream queries: analytical queries (beyond linear aggregates)
- The m-body correlation functions are very popular
  - Requires $O(N^m)$ computational time (N is the number of atoms)
- Of special importance is the Radial Distribution Function (RDF)
  - Often computed as a spatial distance histogram (SDH)
  - 2-body function → $O(N^2)$ time needed for a brute force algorithm

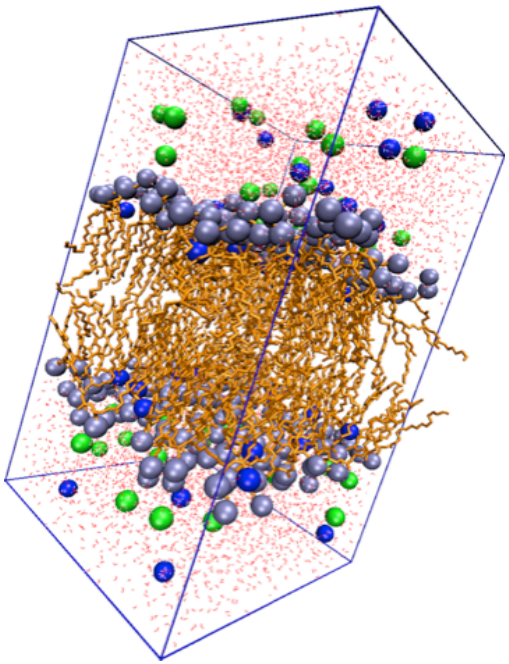


Figure 1. A simulated hydrated dipalmitoylphosphatidylcholine bilayer system.

Problem Statement

Given coordinates of N points, draw a histogram of all pairwise distances - total distance counts will be $N(N-1)/2$

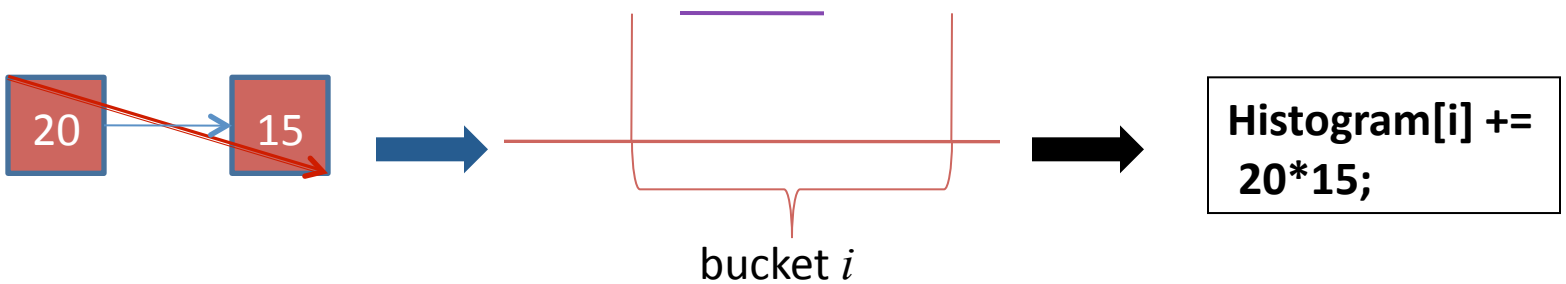
We focus on the standard SDH, in which

- domain of distance $[0, L_{max}]$
- Buckets are of the same width: [0,p), [p, 2p), …
- Query has one single parameter: bucket width p of the histogram, or total number of buckets $l = L_{max}/p$

- Popular simulation packages such as GROMACS[1] all adopt the brute force way of computing SDH
- Can we beat $O(N^2)$?

Our Approach

- Main idea: avoid the calculation of pairwise distances
- Observation: two groups of points can be processed in one shot (i.e., *resolved*) if the range of all inter-group distances falls into a histogram bucket



The DM-SDH algorithm

- Organize all data into a Quad-tree (2D data) or Oct-tree (3D data).
- Cache the atoms counts of each tree node
- Try resolving all pairs of nodes on a tree level M_0
  - If not resolvable, recursively resolve all pairs of children nodes

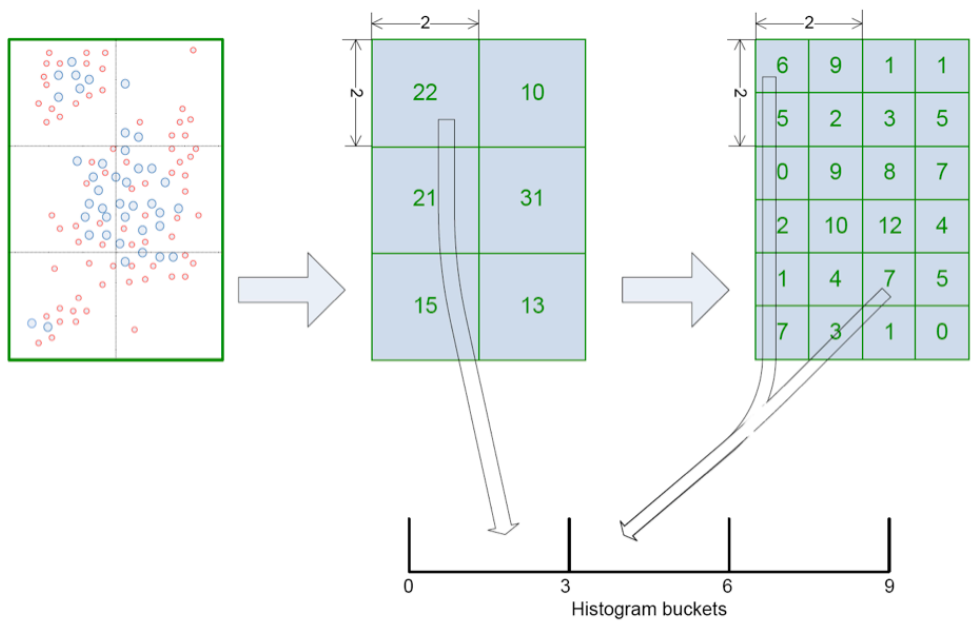


Figure 3. Solving a histogram query (bucket width h = 3) using two density maps (a density map is the counts of all nodes of a whole tree level) generated from raw data (left) with low (middle) and high (right) resolution.

Complexity analysis of DM-SDH algorithm

- Based on a geometric modeling approach
- The main result:

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

1. α(*m*) is the percentage of pairs of nodes that are NOT resolvable on level *m* of the quad(oct)tree.
2. We managed to derive a closed-form for α(*m*)

The above result gives the following analysis

Theorem 1: When the particles are reasonably distributed,
* the time complexity of DM-SDH is $O(N^{(2d-1)/2d})$.

$O(N^{1.5})$ for 2D data and $O(N^{1.667})$ for 3D data
Only in rare cases is the data not reasonably distributed

Approximate Answers

- $O(N^{1.667})$ not good enough for large N?
- Our solution: approximate algorithms based on our analytical model
  - *Time*: Stop before we reach the leaf nodes
  - *Approximation*: for irresolvable nodes, distribute the distance counts into the overlapping buckets heuristically
  - *Correctness*: consult the table we generate from the model

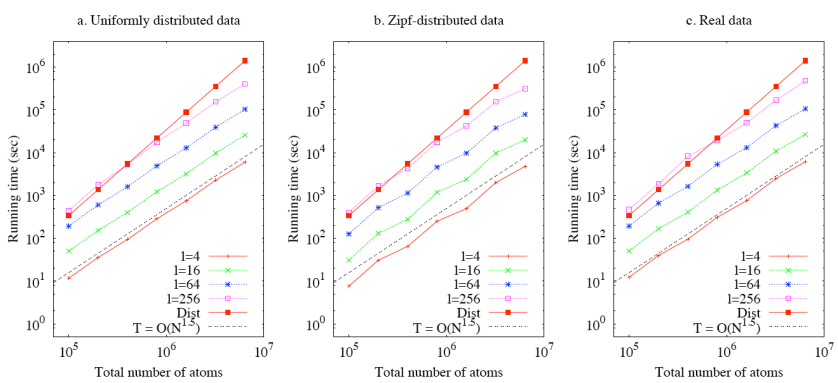


Figure 5. Running time of DM-SDH vs. brute-force algorithm under different 2D data.

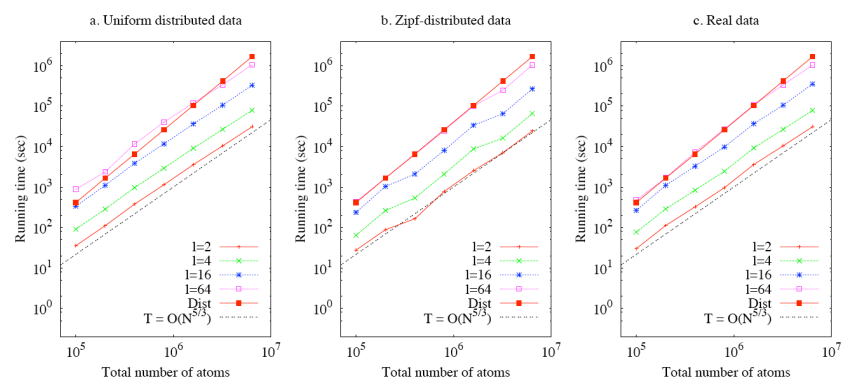


Figure 6. Running time of DM-SDH vs. brute-force algorithm under different 3D data.

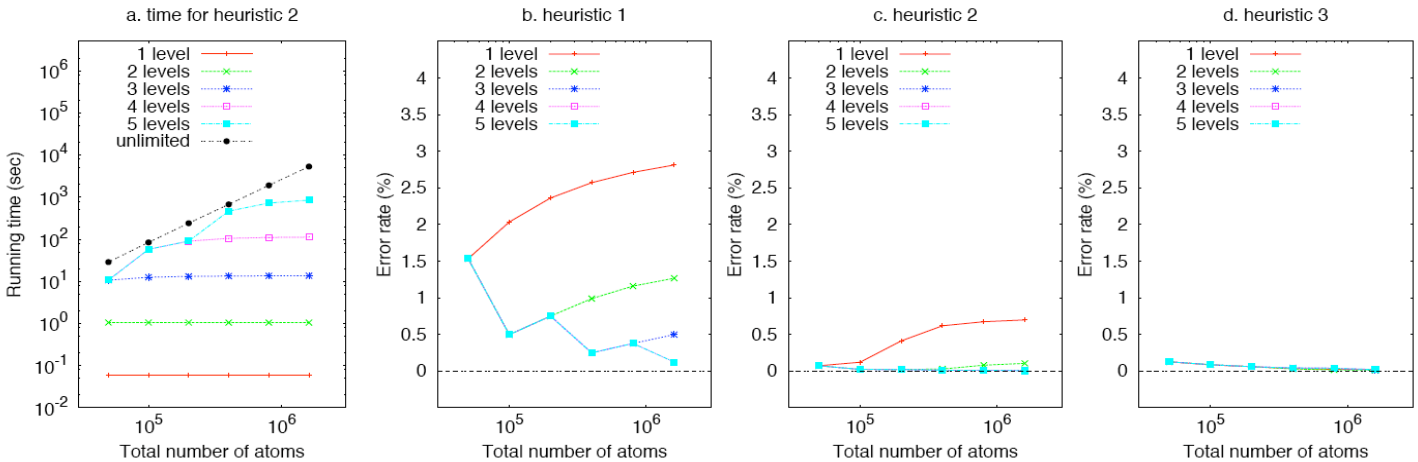


Figure 5. Running time (a) and correctness (b-d) of the approximate algorithm

Summary

- Distance histogram is an important query in simulation databases
- We propose an algorithm based on a quad-tree-based data structure
- Our algorithm outperforms the brute-force approach
- We develop an approximate algorithm with guaranteed error bound and very low time complexity