Available online at www.sciencedirect.com

ScienceDirect

Computer Physics Communications 175 (2006) 339–347

Computer Physics Communications

www.elsevier.com/locate/cpc

Collision-free spatial hash functions for structural analysis of billion-vertex chemical bond networks

Cheng Zhang [a], Bhupesh Bansal [a], Paulo S. Branicio [a,c], Rajiv K. Kalia [a], Aiichiro Nakano [a,*], Ashish Sharma [a,b], Priya Vashishta [a]

[a] *Collaboratory for Advanced Computing and Simulations, Department of Computer Science, Department of Physics & Astronomy, Department of Chemical Engineering & Materials Science, University of Southern California, Los Angeles, CA 90089-0242, USA*

[b] *Department of Biomedical Informatics, Ohio State University, Columbus, OH 43210, USA*

[c] *Departmento de Física, Universidade Federal de São Carlos, São Carlos, SP 13565, Brazil*

Received 1 May 2006; received in revised form 7 June 2006; accepted 13 June 2006

Available online 20 July 2006

Abstract

State-of-the-art molecular dynamics (MD) simulations generate massive datasets involving billion-vertex chemical bond networks, which makes data mining based on graph algorithms such as K-ring analysis a challenge. This paper proposes an algorithm to improve the efficiency of ring analysis of large graphs, exploiting properties of K-rings and spatial correlations of vertices in the graph. The algorithm uses dual-tree expansion (DTE) and spatial hash-function tagging (SHAFT) to optimize computation and memory access. Numerical tests show nearly perfect linear scaling of the algorithm. Also a parallel implementation of the DTE + SHAFT algorithm achieves high scalability. The algorithm has been successfully employed to analyze large MD simulations involving up to 500 million atoms.
© 2006 Elsevier B.V. All rights reserved.

PACS: 07.05.Kf; 07.05.Tp; 61.43.Bn; 61.72.Ff; 82.20.Wt; 89.20.Ff

Keywords: Ring analysis; Topological network; Molecular dynamics simulation; Spatial hash function

1. Introduction

Chemical bond networks are widely used to characterize the structure of materials in solid and liquid phases. Topological analysis of a chemical bond network, e.g., its connectivity and planarity, is essential to understand its physical, chemical and biological properties [1–3]. In such analysis, the structure of a material is abstracted as a topological network or a graph, in which the vertices are atoms and the edges are chemical bonds. Bonds are typically defined between a pair of atoms, for which Pauling's bond order has a value larger than a threshold value or the pair distance is less than a cutoff radius. Most graph-based data mining tools such as SUBDUE [4] and MolFea [5] use such abstractions to study materials and biological systems [6,7]. The popularity of graph mining has soared in recent years [8] in part due to the ever-increasing size and complexity of the chemical networks that are simulated. For such large graphs, structural-analysis algorithms with computational complexity higher than $O(n)$ (n is the number of vertices) are impractical.

Large graph datasets are commonly found in molecular dynamics (MD) simulations, which model materials as a set of atoms, with state-of-the-art MD simulations involving multibillion atoms. Various graph algorithms have been used to analyze MD datasets in the past. An example is ring analysis [9], which has been used to characterize topological order of amorphous materials [10] and to identify and track topological defects such as dislocations [11,12]. Efficient algorithms with near linear scaling are vital for such analysis, especially if the analysis is to be performed in real time during simulation.

In this paper, we present a ring-analysis algorithm that employs dual-tree expansion (DTE) [13] and a spatial hash-function tagging (SHAFT) technique. SHAFT utilizes spatial information on the vertices to design a compact collision-free

[*] Corresponding author.
E-mail address: anakano@usc.edu (A. Nakano).

hash function [14]. The DTE + SHAFT algorithm is parallelized based on spatial decomposition. Numerical tests show nearly linear scaling of the algorithm as a function of the problem size as well as high parallel efficiency. In the next section, we present the DTE + SHAFT algorithm for ring analysis of a graph and its parallelization. Section 3 discusses the results of numerical experiments. Finally discussions and a summary are given in Sections 4 and 5, respectively.

2. Algorithm

To explain our approach, we first define several terminologies and discuss underlying assumptions.

Definition 1 *(Topological network).* A topological network (or graph) $G = (V, E)$ consists of a set of vertices V and a set of edges E, which connect various vertices. A vertex y is a neighbor of x only if they share an edge. In the scope of a chemical bond network, we assume all networks to be undirected without loops [15]. The vertex degree of vertex v in G is the number of edges shared by v with other vertices. A path $p = (v_0, v_1, v_2, \ldots, v_k)$ from v_0 to v_k is a sequence of vertices such that (v_i, v_{i+1}) is an edge in the network, where $0 \leqslant i < k$. The topological distance between two vertices is defined as the length of the shortest path that connects them. A ring in a network is defined as a closed path, in which v_k is equal to v_0 with no other recurrence in the sequence. A ring R is called P-irreducible, if R is the shortest ring containing the path P. There are several criteria for irreducibility, but our algorithm is based on the K-ring criterion proposed by King [9].

Definition 2 *(K-ring).* Given a vertex x and two of its neighbors w and y, a K-ring generated by triplet (w, x, y) is any ring that contains edges (w, x) and (x, y) and has a shortest path (w, y) in $G - x$. Here, if x is a vertex in G, $G - x$ is the network obtained by deleting x from V and all edges containing x from E [15].

A K-ring for a vertex x is the shortest path between two neighboring vertices of x. For example, the paths highlighted by solid lines in Figs. 1(a), 1(b) and 1(c) are K-rings for node x, but the path in Fig. 1(d) is not. This is because the path shown in Fig. 1(d) is not the shortest path between the two neighbors of x, i.e., vertices P and R. Therefore, vertex X in Fig. 1 has only 3 four-member K-rings.

It is important to note that a given vertex may have K-rings of different sizes depending on the choice of the two neighbor vertices. For example, vertex X in Fig. 2 has three K-rings of varying lengths (ring a of length 6, ring b of length 4, and ring c of length 5). The purpose of K-ring analysis is thus to analyze K-ring statistics, i.e., the histogram of the lengths of all K-rings of all vertices in a network.

A K-ring containing L vertices is often called an *L-fold ring* [9]. However, for binary ionic compounds such as silicon carbide, where bonds between the same elements are discarded, one fold (member) is typically defined as one (A, B) segment, where A and B are atomic species (e.g., A = silicon and B = carbon in silicon carbide) [10,16]. In this case, a L-fold (L-member) ring actually contains $2L$ vertices. To avoid ambiguity, we will only refer to a K-ring with L vertices as an L-member ring throughout the paper. The number of L-member rings in the network is unique to a particular crystalline structure and it is possible to state the nature of the connectivity of the elementary units, beyond the nearest neighbor, in terms of these L-member rings. Non-crystalline and disordered networks may also be characterized in terms of their ring structure profile, by analyzing the K-ring length distribution. Compared to the simple case of perfect crystalline systems, more extensive study has been conducted on disordered net-

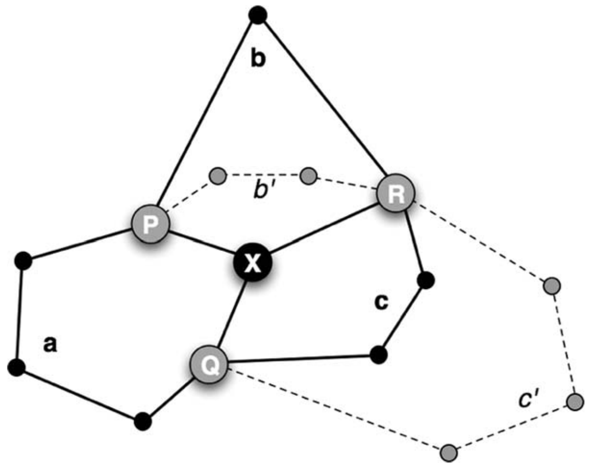


Fig. 2. K-ring statistics for vertex x in an network. Only the paths labeled as a, b and c are K-rings. Paths b' and c', though closed and without any cycles, are not K-rings, because they are not the shortest paths between the corresponding neighboring vertices of x. Vertex x has 3 K-rings a, b, c of lengths 6, 4, and 5, respectively.

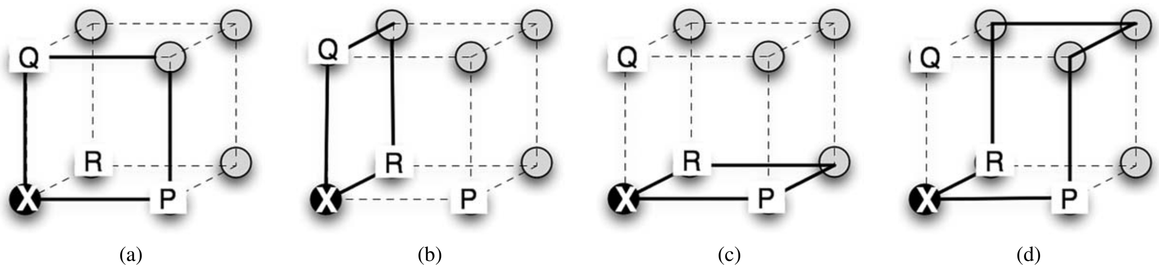


Fig. 1. K-rings emanating from vertex x in a simple cubic structure, in which the edges are denoted by solid or dashed lines. Only the paths shown with solid lines in (a), (b) and (c) are considered as K-rings. The path in (d), though closed and without any cycles, is not a K-ring, because it is not the shortest path between the neighboring vertices P and R.

works such as silica glass [9,10,16–18]. Ring analysis is also an effective tool to study the structure change in dynamically loaded systems [11,12]. Since material processes such as dislocation motions and fracture result in changes to a material's structure, the composition of these statistics may change for all constituent vertices. The rate of change and its location thus provide information about the transformations that are taking place in the system and can also be used as an indicator of topological anomaly when used in conjunction with other physical attributes such as stress or temperature.

Definition 3 *(Hash function).* A hash function is a mapping from each element of an input set X to an element in a hash key value set K, where $|K| \ll |X|$ ($|K|$ and $|X|$ are the numbers of elements in sets K and X, respectively). Hash functions are widely used in encryption of digital signature systems and address indexing of a table's elements.

A hash collision occurs when two distinct inputs are mapped to the same hash key value. Collision is a common problem for hash functions as the mapped configuration space is usually smaller than the initial configuration space. Techniques such as hierarchical hash tables and linked lists as hash keys are used to minimize hash collisions.

Although our Definition 1, of topological network, conforms to previous publications on ring analysis, the actual model we use has a different level of abstraction. In addition to topological data, our spatial hash function also requires the Euclidean positions of vertices as an input. In MD simulations, this information is readily available at no extra computational cost or storage space, since the coordinates of all vertices (i.e., atomic positions) are maintained and updated per MD iteration step. On the other hand, determination of a chemical bond, i.e., an edge in the network, typically requires extra computation. For example, a bond is often defined as a vertex pair whose Euclidean distance is less than a given cutoff radius. Instead of requiring bond data as an input, our program takes only the vertex coordinates as an input, so that it is more storage-efficient. It performs $O(n)$ computation to recover the bond data.

In order to search for K-rings, we need the adjacencies, i.e., the bonding information, of all the vertices in the network. We use an adjacency list to store each vertex's degree and neighbor indices instead of an adjacency matrix for better memory efficiency. For simulations of chemical bond networks, the data saved are usually coordinates of the vertices, where the only criterion for bonding is the inter-vertex Euclidean distance. For networks with million-to-billion vertices, building the adjacency list incurs non-trivial computational cost. The naïve approach would calculate $O(n^2)$ Euclidean distances between all vertex pairs. Instead, we partition the Euclidean space into spatial grids, where the dimension C_i $(i = x, y, z)$ of the grid satisfies $C_i \geqslant r_{\text{cutoff}}$ (r_{cutoff} is the cutoff length of chemical bonds). This spatial subdivision scheme limits the range of Euclidean distance calculations to a vertex's neighboring grids, where the vertices within a grid cell are traversed using a neighbor cell list. This approach reduces the computational complexity from $O(n^2)$ of the pair-wise Euclidean distance comparison to $O(n)$.

With the adjacency information thus available, K-ring analysis is accomplished through a Breadth-First Search (BFS) algorithm such as the one used by Rino et al. [10]. We refer to this procedure as Single-Tree Expansion (STE) algorithm, since it expands only one connectivity tree at a time. For a given node x, two of its neighbors w and y are selected to construct a K-ring. The STE algorithm starts from w and recursively expands its neighbors in a BFS fashion until it finds the shortest path to reach y without visiting x. This shortest path plus segment (w, x, y) form a K-ring chosen for the (w, x, y) combination. To identify rings of up to L members, a depth cut-off of $L - 2$ expanding from w suffices and limits the computational cost of BFS to $O(k^{L-2})$, where k roughly equals to the average vertex degree minus 1. The total time complexity for all combinations of x, w and y in an n-vertex network is $O(k^{L-2}) \cdot O(k^2 n) = O(k^L n)$. However, for large systems with millions of vertices per processor, this algorithm often fails to return results in a reasonable time. There are other algorithms that calculate the shortest paths for vertex pairs using Dijkstra's algorithm [19]. But their computational and memory efficiency are also limited for large chemical bond networks with millions of vertices.

To address this difficulty, we have designed a Spatial Hash-Function Tagging (SHAFT) algorithm built upon a Dual-Tree Expansion (DTE) algorithm to improve the efficiency of ring analysis for large systems.

2.1. Dual-tree expansion (DTE) algorithm

If vertex p is in the same L-member ring with x, the shortest path from x to p has to be no longer than $L/2$. When the BFS algorithm searches from vertex w for $L - 2$ steps, some trial paths visit vertices that are too far from x to be possible candidates. In order to avoid these wasteful searches, for triplet (w, y, x), we expand the connectivity trees from w (the left tree) and y (the right tree) at the same time, instead of searching for paths from w to y. The vertices are expanded in the BFS fashion until a desired depth is reached or a closed path is identified. A vertex's neighbors unvisited in previous depths are inserted as its children in the tree. If leaf vertex p in the w-rooted tree and leaf vertex p' in the y-rooted tree are identical, while sharing no intermediate vertices in their paths, a ring has been identified. The ring can be dissected into three segments: (w, x, y), (w, p), and (y, p), where $p' = p$. When compared to the original algorithm, this method reduces the number of vertices searched from approximately d^L to $d^{L/2}$, where d is a constant no larger than the average number of vertex degrees minus 1, and L is the length of the ring. This scheme can be further optimized for rings with only even number of vertices, which is a valid assumption for binary ionic bond networks described previously. In the even-vertex case, we only need to compare the leaf vertices on the same depth from the left and right trees, which reduces the computational complexity of $O(d^{L_{\max}})$ to $O(d^{L_{\max}/2})$ for searching up to $L_{\max}$-member rings from a single vertex (see Fig. 3). The resulting dual-tree expansion (DTE) algorithm is shown in Table 1.

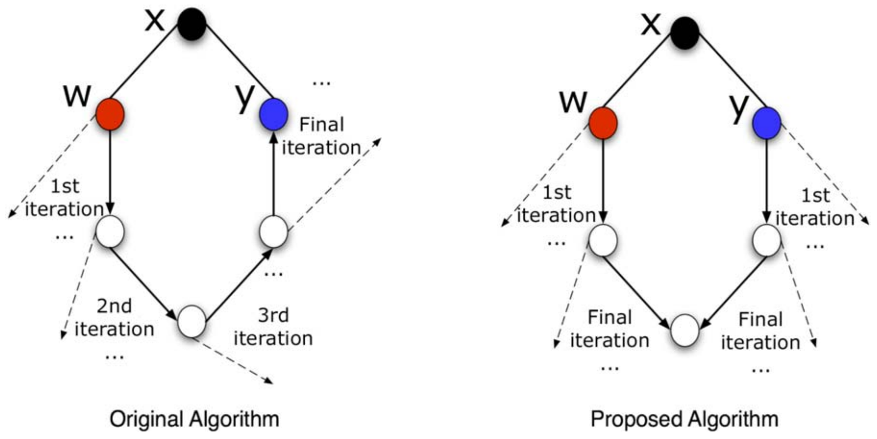


Fig. 3. Comparison of the proposed dual-tree expansion (DTE) algorithm with the original algorithm using a 6-member ring as an example. The depth of search in the proposed algorithm is reduced by half.

Table 1
Dual-tree expansion algorithm

```
Algorithm dual_tree_expansion()

Input:
    $V$ = Set of all vertices (i.e., atoms)
    $R_c$ = Ring cutoff range (Euclidean)
    $R_{bc}$ = Bond cutoff distance (Euclidean)
    $L_{MAX}$ = Maximum length of ring (integer)
Output:
    The K-ring statistics for all vertices in the network
Variables:
    Neighbors($V$) = Set of vertices that share an edge with vertex $V$
    $K_v(L)$ = Number of $L$-member rings that go through vertex $V$
    $L_{ij}$ = Length of the ring formed with path $(V_i, V, V_j)$
Steps:
    0   create adjacency list $G$ for all node in $V_p$ using $R_{bc}$ as cutoff distance
    1   for every vertex $V \in V_p$
         for each vertex pair $V_i$ and $V_j$ in Neighbors($V$) do
                $A_1 = \{V_i\}$
                $A_2 = \{V_j\}$
                $L_{ij} = 0$
                while ($A_1 \cap A_2 = \emptyset$ AND $L_{ij} < L_{MAX}$) do
                        $L_{ij} = L_{ij} + 2$
                        if ($A_1 \cap$ Neighbors($A_2$) $\neq \emptyset$ OR $A_2 \cap$ Neighbors($A_1$) $\neq \emptyset$)
                            $L_{ij} = L_{ij} + 1$
                            break
                        else if (Neighbors($A_1$) $\cap$ Neighbors($A_2$) $\neq \emptyset$)
                            $L_{ij} = L_{ij} + 2$
                        $A_1$ = Neighbors($A_1$)
                        $A_2$ = Neighbors($A_2$)
                if ($L_{ij} < L_{MAX}$)    $++K_V(L_{ij})$
```

2.2. *Spatial hash-function tagging (SHAFT) algorithm*

The above algorithm is optimal in terms of the number of vertices expanded. However, still a great amount of time is spent to match the leaf vertices from the left and right trees to identify a closed path. For example, in a system with average vertex degree of 6, there could easily be more than 10^3 vertices on the eighth level of a connectivity tree (after excluding previously visited vertices, a reduced vertex degree of 3 would give an estimated $3^7 = 2187$), which requires more than $1000^2 = 10^6$ comparisons and conditional operations. The number of comparisons can be reduced to $O(k \log k)$ by sorting the vertices, where k is the average number of leaf vertices. But it still makes up a large portion of the computation when the trees grow up to as many as 10 levels (which is common in disordered system), as k increases exponentially with the tree height. An alternative approach avoids pair comparison by employing a vertex table, where vertices visited on the leaf level are flagged and any double-flagged vertex implies a closed path. This method saves computation but costs more memory to store all the vertices in the table. Cache misses when referencing and updating the table can pose a serious problem in the light of multilevel memory hierarchies. Absence of locality of edges in the adjacency matrix is common in MD simulations, where atoms diffuse over time and its vicinity can be randomly rearranged. When dealing with systems with million-to-billion vertices, cache misses can become the major bottleneck in achieving real-time speed. The tradeoff thus is either heavy computation (conditional operations) or frequent cache misses. Cache-oblivious models and other techniques have been developed to gain better cache performance when dealing with massive scientific data [20–22].

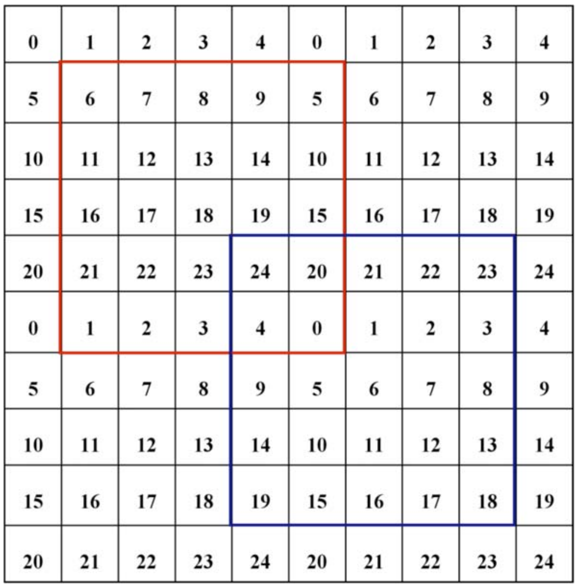
We propose an alternative solution using a spatial hash-function tagging (SHAFT) algorithm, which has the following features:

- It uses a hash table to avoid pair comparison of leaf vertices. Each vertex is mapped to an element in the table according to its spatial information.
- The table size is small and independent of $|V|$ in the network.
- It is guaranteed that there is no collision in the range of the hash table when calculating ring structures from any vertex.

Normally, a hash function is prone to collision when its configuration space is compressed, and thus the second and third features contradict in general. In the SHAFT algorithm, on the other hand, the hash function is collision-free only locally within the scope of interest instead of globally, i.e., no two vertices shall have the same integer hash value within the ring cutoff range from any vertex. This is achieved by exploiting the correlation between the vertex coordinates, in the form of upper and lower bounds for all inter-vertex distances. One feature of a chemical bond network is that the distance-dependent repulsive force between any two adjacent atoms (vertices) prevents them from approaching too close. On the other hand, the chemical bond (connectivity or edge) between them breaks when they are apart by more than a certain length. We denote the upper bound for inter-vertex distance as r_{upper} and the lower bound as r_{lower}. Within any cube of edge $b = r_{\text{lower}}/\sqrt{3}$ there cannot be more than one vertex, because the diagonal length of the cube is the maximum distance allowed. Therefore, the integer cube index can be used as a unique hash value without collision. On the other hand, the upper bound determines how far to search for a vertex's neighbor. Along with $L_{\max}$, the maximum length of a ring defined by user, it gives the maximum Euclidean distance $r_{\text{upper}}(L_{\max}/2)$ between any two atoms in a ring. It also implies that all vertices in the same ring with vertex x must lie within a cube of side length $c = r_{\text{upper}}L_{\max}$ centered at x.

The actual hash value is constructed as follows. First note that no two integers out of m consecutive integers have the same remainder when divided by m (i.e., they are not congruent modulo m). Similarly, for any $m \times m \times m$ sub-region in a large 3D grid, no two grid points have the same set of indices modulo m (see Fig. 4).

Now let the cell size b equal to $r_{\text{lower}}/\sqrt{3}$ in the grid and $m = \lceil c/b \rceil = \lceil r_{\text{upper}}L_{\max}/(r_{\text{lower}}/\sqrt{3})\rceil$. Then, there can be no more than one vertex in each cell, i.e., all vertices have unique cell indices. Since all cells have unique indices modulo m in any m^3 sub-domain, all vertices in the sub-domain have unique cell indices modulo m.



0	1	2	3	4	0	1	2	3	4
5	6	7	8	9	5	6	7	8	9
10	11	12	13	14	10	11	12	13	14
15	16	17	18	19	15	16	17	18	19
20	21	22	23	24	20	21	22	23	24
0	1	2	3	4	0	1	2	3	4
5	6	7	8	9	5	6	7	8	9
10	11	12	13	14	10	11	12	13	14
15	16	17	18	19	15	16	17	18	19
20	21	22	23	24	20	21	22	23	24

Fig. 4. An example of the collision-free spatial hash function in 2D. Within any window no larger than the hash function's modulus, there will be no two identical numbers.

After each vertex is assigned a hash value, this new index is saved along with its global index in the adjacency list, which enables fast fetching of this tag. When a vertex is inserted into the tree in the DTE algorithm, the corresponding element in the hash table of size m^3 is flagged. The same operation on the global array in the original DTE algorithm can be applied to this hash table to detect closed paths. Due to the small and constant size of the hash table and therefore less cache misses in large systems, this scheme provides better scalability compared to the original DTE algorithm. The SHAFT algorithm is shown in Table 2, where the ceiling function $\lceil x \rceil$ is the smallest integer that is greater than x, the floor function $\lfloor x \rfloor$ is the largest integer that is less than x, and % denotes modulo operator.

The following example illustrates the typical size of the hash table: r_{lower} in most materials is larger than 1.4 Å, and thus the unit grid length $b = R_{\text{lower}}/\sqrt{3} = 1.4/\sqrt{3} = 0.808$. For a 10-member ring, we can take the ring computation cutoff $c = R_{\text{bc}}L_{\text{MAX}} = 15$ Å, so that $m = \lceil c/b \rceil = 20$. In three dimensions, the corresponding mapped configuration space is 8,000, which is greatly reduced from the original configuration space of millions and fits in an L2 cache. The mapped configuration space is even smaller for rings of (A, B, A, B) type in binary compounds. Given the current vertex species, we always know which species the next vertex on the ring should be. Therefore two vertices are allowed per cell, one for each species, as they can be distinguished by the hash value plus the species index. In this case we can replace r_{lower} by the minimum distance between vertices of same species, which is usually much larger than r_{lower}. In most ionic compound materials, distances between vertices of same species are usually larger than 2 Å, which produces $b = 1.154$. For $c = 15$ Å, $m = 13$ and the configuration space size of the hash function is only $13^3 = 2{,}197$.

Table 2
Spatial hash-function tagging algorithm

```
Algorithm spatial hash function tagging (SHAFT) + DTE
Input:
    $V =$ Set of all vertices (i.e., atoms)
    $C(V) =$ 3D coordinates of all vertices
    $R_{\text{bc}} =$ Bond cutoff distance (Euclidean)
    $R_{\text{lower}} =$ Minimum distance between vertices
    $L_{\text{MAX}} =$ Maximum length of ring (integer)
Output:
    The K-ring statistics for all vertices in the network
    List of atoms with abnormal ring profile
Variables:
    Neighbors$(V) =$ Set of vertices that share an edge with vertex $V$
    $K_v(p) =$ Number of $p$-member rings that go through vertex $V$
    $L_{\text{depth}} =$ Current depth of the expanded tree (both left and right)
    $L_{\min}(V, V_n) =$ Minimum number of steps to reach vertex $V$ from root vertex through $V_n$
    $L_{ij} =$ Length of the ring formed with path $(V_i, V, V_j)$
Steps:
    0   build the spatial hash function table and integrate into the adjacency list $G$ (as secondary reference)
        for each vertex $V \in V_{\text{p}}$
            for each spatial dimension $i \in \{x, y, z\}$
                $q_i = \lfloor C_i(V)/b \rfloor$ where $b = R_{\text{lower}}/\sqrt{3}$
                $q_i = q_i$ modulo $m$ where $c = R_{\text{bc}} L_{\text{MAX}}$ and $m = \lceil c/b \rceil$
                $q(V) = q_3 \times m^2 + q_2 \times m + q_1$
    1   for every vertex $V \in V_{\text{p}}$
         for each vertex pair $V_i$ and $V_j$ in Neighbors$(V)$ do
                $A_1 = \{V_i\}$
                $A_2 = \{V_j\}$
                $L_{\text{depth}} = 1$
                $L_{\min}(q(A_1), V_i) = L_{\text{depth}}$
                $L_{\min}(q(A_2), V_j) = L_{\text{depth}}$
                while $(\{A \mid L_{\min}(q(A), V_i) < \infty \ \& \ L_{\min}(q(A), V_j) < \infty\} = \emptyset$ AND $L_{\text{depth}} < L_{\text{MAX}}/2)$ do
                        $L_{\text{depth}}$ ++
                        $A_1 =$ Neighbors$(A_1)$
                        $A_2 =$ Neighbors$(A_2)$
                        $L_{\min}(q(A_1), V_i) = L_{\text{depth}}$
                        $L_{\min}(q(A_2), V_j) = L_{\text{depth}}$
                if $(L_{\text{depth}} < L_{\text{MAX}}/2)$ $L_{ij} = L_{\min}(q(A), V_i) + L_{\min}(q(A), V_j)$
```

Note that the ring cutoff range $c = r_{\text{upper}} L_{\max}$ is a conservative estimate, assuming the ring consists of two parallel and straight chains. In real cases, however, the ring cutoff range is much smaller because the bond angles (i.e., the angles between two consecutive edges) are usually less than 180°. One can construct a tighter bound for the ring cutoff range using specific information on the bond angle distribution, which will further reduce the mapped configuration space.

2.3. Parallelization

The ring analysis algorithm has been implemented on parallel computers based on spatial decomposition [23], in which the physical Euclidean space is divided into subspaces of equal volume Ω that are assigned onto compute nodes in a parallel computer. The compute nodes are logically arranged as a 3D mesh of size $P = P_x \times P_y \times P_z$. Each node is responsible for calculating the ring structure of local vertices (i.e., atoms). However, those vertices near the subspace boundary need information from neighboring subspaces to complete the calculation. So a skin of thickness c and volume $\sim c\Omega^{2/3}$ on each compute node is copied to the neighboring nodes before the ring analysis takes place. The thickness c is equal to the ring cutoff range and depends on the user-defined ring length $L_{\max}$. Each node calculates the local ring structure independently after the skin copy. The parallelization efficiency η can be estimated to be $1 - t_{\text{comm}}/t_{\text{comp}}$, where t_{comp} is the computing time and t_{comm} is the communication time. Assuming even vertex density, t_{comp} scales linearly with Ω and t_{comm} scales with skin volume $c\Omega^{2/3}$, leading to an $O(\Omega^{-1/3})$ scalability for $t_{\text{comm}}/t_{\text{comp}}$. When Ω is large, the efficiency η is approaching 100% due to large volume-surface ratio of the subsystems.

3. Numerical results

We have performed numerical tests to compare three algorithms: STE, DTE, and DTE combined with SHAFT. The test used a PC with dual Intel Xeon 2.8 GHz CPUs with 2 GB of memory. We evaluate the efficiency of the three algorithms in three categories—clock time, number of instructions and num-

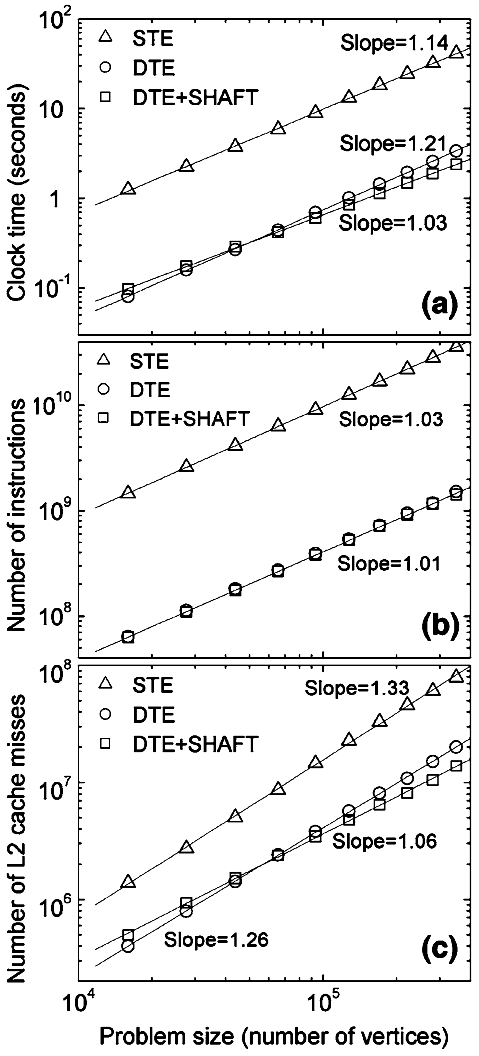


Fig. 5. (a) Log–log plot of clock time vs. problem size, where the DTE combined with SHAFT is compared against DTE alone and STE. DTE with SHAFT gives the best performance for large problem size and scales roughly linear. Lines are linear fits with slopes 1.14, 1.21, and 1.03 for STE, DTE, and DTE + SHAFT algorithms, respectively. (b) Number of instructions vs. problem size for the three algorithms in a log–log plot. Lines are least-square fits with slopes 1.03, 1.01, and 1.01 for STE, DTE, and DTE + SHAFT algorithms, respectively. (c) Log–log plot of cache misses vs. problem size for STE, DTE and DTE + SHAFT. Lines are least-square fits with slopes 1.33, 1.26, and 1.06 for STE, DTE, and DTE + SHAFT algorithms, respectively.

ber of cache misses—for chemical networks of sizes ranging from 10^4 to 3×10^5 (number of vertices). The maximum ring cutoff is chosen to be 6 for simplicity and quick results.

Fig. 5 compares the execution time (Fig. 5(a)), the number of instructions (Fig. 5(b)), and the number of cache misses (Fig. 5(c)) of the three algorithms as a function of the number of vertices. For systems of about 3×10^5 vertices, the DTE algorithm outperforms the STE algorithm by a factor of 15 or more. For systems of about 300 thousand vertices, the DTE algorithm outperforms the STE algorithm by a factor of 15 or more. This speedup is mainly attributed to the reduction of instructions, as indicated by similar gain ratios in clock time (Fig. 5(a)) and number of instructions (Fig. 5(b)). However, this speedup ratio dwindles for larger problem size as the computing time of DTE scales roughly as $O(N^{1.21})$ while STE as $O(N^{1.14})$. This is because DTE algorithm employs an array of size N to discover common leaf nodes for each vertex, and reading/writing this array with poor data locality cause cache misses of $O(N^{1.26})$. The lower proportion of total clock time retired on instructions in DTE than in STE implies that the cache misses play a bigger role for DTE performance. However, SHAFT algorithm is able to reduce the computing time scalability to almost perfectly linear by using a constant-sized leaf node hash table. The overhead eliminated by SHAFT is basically cache misses instead of instructions. This can be verified by the correlation between the clock time (Fig. 5(a)) and number of cache misses (Fig. 5(c)), whereas the number of instructions is roughly unchanged by introducing SHAFT. Another notable phenomenon is the crossover of the two DTE algorithms in clock time and cache misses. This implies that the spatial hash table we introduce in SHAFT becomes an overhead in small networks, where the hash table size is comparable to the total problem size. Consequently DTE with SHAFT shows advantage only in large-scale problems ($N > 10^5$). For larger ring cutoff, we expect greater efficiency boost of DTE + SHAFT from both DTE and STE.

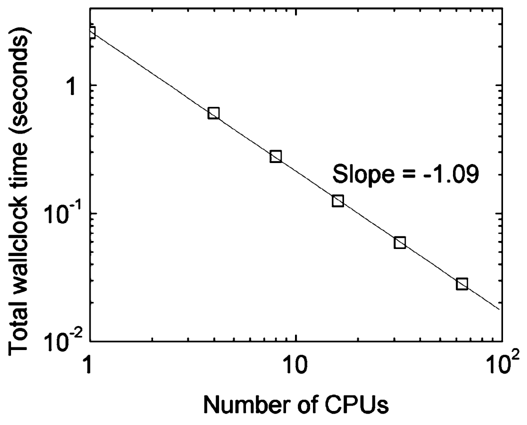


Fig. 6. Execution time of the DTE + SHAFT algorithm as a function of the number of computing nodes with a fixed problem size (5×10^5 vertices). The line is the least square fit with slope −1.09.

We have also performed benchmark tests of the parallel DTE + SHAFT algorithm on a Linux cluster of dual Intel Xeon 2.8 GHz CPUs and 2 GB of memory per node. The total problem size is held fixed at 5×10^5 vertices. The strong scalability test gives a roughly unit slope in the log–log graph (Fig. 6). Superlinear scalability is observed in this plot, which implies greater overall dependence on the cache size. In addition, the communication time alone has also exhibited super scalability, which indicates that communication efficiency is also influenced by the cache size.

4. Discussion

The new algorithm has enabled ring analysis of massive datasets from 200–500 million-atom MD simulations of hypervelocity impact damage of advanced ceramics (aluminum nitride [12], silicon carbide, and alumina), 19-million-atom simulation of indentation damage of superhard nanocrystalline

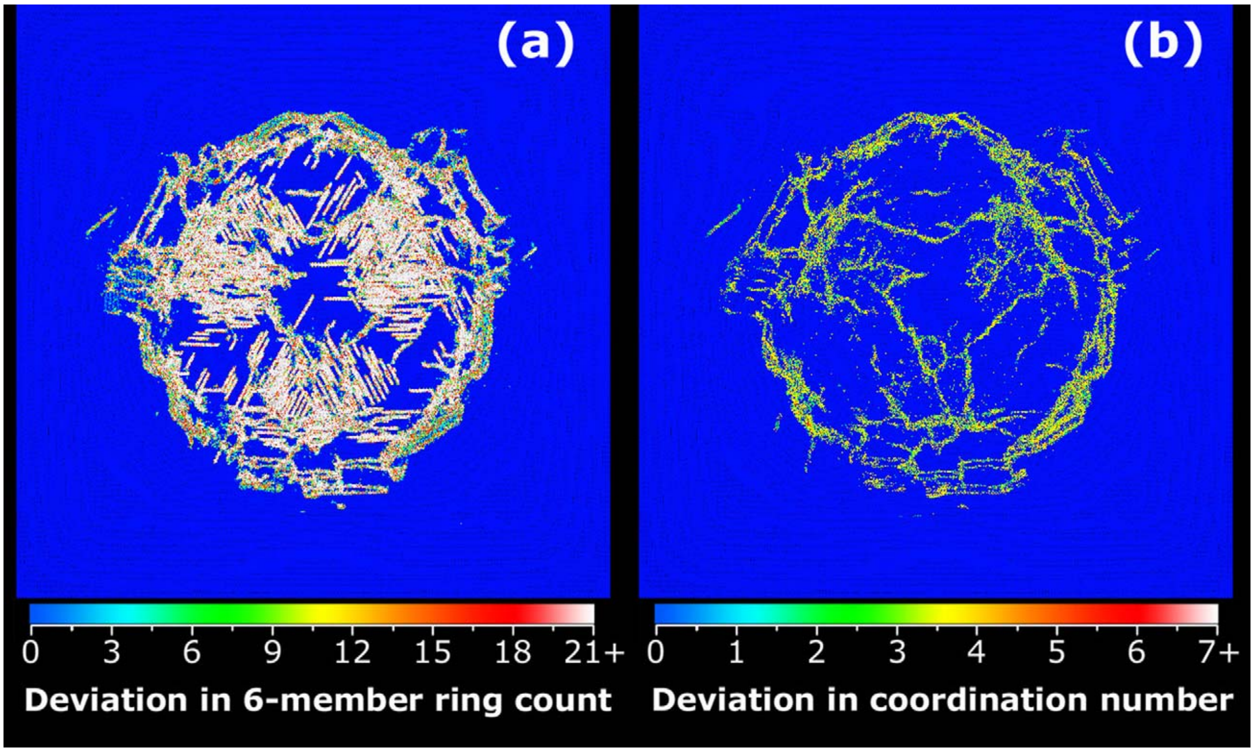

(a)
(b)
0 3 6 9 12 15 18 21+
0 1 2 3 4 5 6 7+
Deviation in 6-member ring count
Deviation in coordination number

Fig. 7. (a) A thin slice of a 500 million-atom alumina target 40 nm in front of the projectile during hypervelocity impact simulation. Deviation in the number of 6-member rings from perfect crystalline atoms (blue) is color-coded using the gradient bar above. (b) The same plane colored by deviation in coordination number from perfect crystalline atoms (blue). (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

silicon carbide (n-SiC) [11], and 15-million-atom simulation of fracture in amorphous silica [24]. On these datasets, the DTE algorithm gains a speedup of 10 or more against STE, while the SHAFT implementation gives an additional edge of 1.5–2.

For hypervelocity impact simulation of alumina (Al_2O_3) consisting of 500 million atoms on 960 AMD Opteron processors, it takes less than a second to identify all the disordered atoms due to different deformation mechanisms using the DTE + SHAFT algorithm. In a perfect alumina crystal, each aluminum atoms has 12 unique 6-member rings, any deviation in the ring profile from this indicates topological disorder. Combined with coordination number analysis, the ring algorithm can distinguish between deformation mechanisms such as twinning, slip, amorphization, and structural transformation. In a snapshot of a thin slice of material 40 nm in front of the projectile in Fig. 7(a), ring analysis reveals rhombohedral twinnings (colored as white) forming in three possible orientations within a circle of pyramidal slips (colored as red or green), as both types of deformation give different ring numbers from normal crystalline atoms (colored as blue). While pyramidal slips cause deviation in coordination number as well (see Fig. 7(b)), rhombohedral twinnings can effectively be identified only by ring analysis. The DTE + SHAFT algorithm has enabled us to globally locate damages, especially rhombohedral twinnings, in the 500-million atom simulation on 960 Opteron processors in less than 0.5 second. This was not possible in a realistic time frame using the naïve algorithm due to insufficient physical memory.

5. Summary

As the size of chemical bond network studied grows proportionally with the fast-increasing computing power, the original structure analysis algorithms that scale as $O(n^k)$ (with $k > 1$) is out-paced. We propose a new real-time algorithm that improves both computing complexity and data locality, and as a result the combined speedup in the overall efficiency. The lower computing complexity is achieved by using dual-tree expansion (DTE) algorithm and the better data locality through a spatial hash-function tagging (SHAFT) algorithm. The first approach gives a speedup factor of 15 or more for the problem size of our interest, while the second scheme provides an additional improvement of 40%–50%. The SHAFT algorithm can also be applied to other general structural analysis of chemical bond networks.

Acknowledgements

This work was partially supported by AFOSR-DURINT, ARO-MURI, DOE, and NSF. Numerical tests were performed at the University of Southern California using the 3600-processor Linux cluster at the Research Computing Facility and the 1600-processor Linux clusters at the Collaboratory for Advanced Computing and Simulations.

References

[1] E. McCafferty, Corrosion Science 44 (2002) 1393.

[2] S.J. Haggarty, P.A. Clemons, S.L. Schreiber, Journal of the American Chemical Society 125 (2003) 10543.
[3] J.C. Nacher, N. Ueda, T. Yamada, M. Kanehisa, T. Akutsu, BMC Bioinformatics 5 (2004) 207.
[4] D.J. Cook, L.B. Holder, IEEE Intelligent Systems & their Applications 15 (2000) 32.
[5] R. Luc De, S. Kramer, The levelwise version space algorithm and its application to molecular fragment finding, in: Proceedings of the 17th International Joint Conference on Artificial Intelligence, Seattle, WA, USA, 2001.
[6] D.J. Cook, L.B. Holder, S.B. Su, R. Maglothin, I. Jonyer, IEEE Engineering in Medicine and Biology Magazine 20 (2001) 67.
[7] K. Stefan, R. Luc De, H. Christoph, Molecular feature mining in HIV data, in: Proceedings of the Seventh ACM SIGKDD International Conference on Knowledge discovery and Data Mining, ACM Press, San Francisco, CA, USA, 2001.
[8] T. Matsuda, H. Motoda, T. Washio, Advanced Engineering Informatics 16 (2002) 135.
[9] S. King, Nature 213 (1967) 1112.
[10] J.P. Rino, I. Ebbsjo, R.K. Kalia, A. Nakano, P. Vashishta, Physical Review B 47 (1993) 3053.
[11] I. Szlufarska, A. Nakano, P. Vashishta, Science 309 (2005) 911.
[12] P.S. Branicio, R.K. Kalia, A. Nakano, P. Vashishta, Physical Review Letters 96 (2006) 065502.
[13] Similar techniques have been used previously, see, e.g. X.L. Yuan, A.N. Cormack, Computational Materials Science 24 (2002) 343.
[14] T.H. Cormen, C.E. Leiserson, R.L. Rivest, C. Stein, Introduction to Algorithms, second ed., MIT Press, Cambridge, MA, 2001.
[15] D.S. Franzblau, Physical Review B 44 (1991) 4925.
[16] L. Guttman, Journal of Non-Crystalline Solids 116 (1990) 145.
[17] C.S. Marians, L.W. Hobbs, Journal of Non-Crystalline Solids 124 (1990) 242.
[18] L.W. Hobbs, C.E. Jesurum, V. Pulim, B. Berger, Philosophical Magazine A 78 (1998) 679.
[19] X.L. Yuan, A.N. Cormack, Computational Materials Science 24 (2002) 343.
[20] M. Frigo, C.E. Leiserson, H. Prokop, S. Ramachandran, Cache-oblivious algorithms, in: Proceedings of the 40th Annual Symposium on Foundations of Computer Science, New York, NY, USA, 1999, p. 285.
[21] E.D. Demaine, Cache-oblivious algorithms and data structures, in: Lecture Notes of the EEF Summer School on Massive Data Sets, University of Aarhus, Denmark, 2002.
[22] J.S. Vitter, ACM Computing Surveys 33 (2001) 209.
[23] R.K. Kalia, W. Jin, S.W. Deleeuw, A. Nakano, P. Vashishta, International Journal of Quantum Chemistry (1993) 781.
[24] Z. Lu, K. Nomura, A. Sharma, W. Wang, C. Zhang, R.K. Kalia, N. Aiichiro, P. Vashishta, Physical Review Letters 95 (2005) 135501.