ELSEVIER

Computer Physics Communications 119 (1999) 135–148

Computer Physics Communications

www.elsevier.nl/locate/cpc

Near-neighbor calculations using a modified cell-linked list method

William Mattson [1], Betsy M. Rice [2]

The U.S. Army Research Laboratory, Aberdeen Proving Ground, MD 21005, USA

Received 29 September 1998; revised 23 November 1998

Abstract

We have modified the conventional cell-linked list method to reduce the number of unnecessary internuclear distance calculations in molecular simulations of systems containing many particles. In the conventional method, the simulation space is partitioned into cells with edge lengths no less than the cutoff distance of the interaction potential (r_{cut}). The atoms are assigned to cells according to their spatial positions, and all internuclear distances for atoms within a cell and atoms in the same and nearest neighbor cells are evaluated. While this method ensures that the internuclear separations between all atom pairs within r_{cut} are calculated, it allows for unnecessary internuclear distance calculations between pairs that are within the volume encompassing the neighbor cells, but that are separated by more than r_{cut}. The modified method presented here allows for reductions in the cell sizes and the number of atoms within the volume encompassing the neighbor cells. These reductions decrease the number of atoms that are outside of the interaction range and the number of unnecessary internuclear distance calculations while ensuring that all internuclear distances within the cutoff range are evaluated. We present algorithms to determine the volume with the minimum number of neighbor cells as a function of cell size and the identities of the neighboring cells. We also evaluate the serial performance using the modified form as functions of cell size and particle density for comparison with the performance using the conventional cell-linked list method. Published by Elsevier Science B.V.

PACS: 31.15.Qg; 33.15.Dj

1. Introduction

Popular molecular simulation techniques such as molecular dynamics or Monte Carlo are used to study the physical and chemical processes occurring in systems containing large numbers of atoms at the atomic level [1]. These methods require evaluation of either the total potential energy of a system of N atoms (V_{Tot}) or the gradients of the potential energy. The total potential energy consists of terms that describe the various interactions among the atoms in the system. These terms are usually functions of internal coordinates, such as internuclear distances between two atoms, bond angles among three atoms, or torsional angles among four atoms. For condensed phase modeling, the total potential energy is often described as a sum of two-body interactions over all atom pairs. The interaction terms are typically simple functions of the internuclear distance r_{ij} between atoms i and j,

[1] Current mailing address: Department of Physics, University of Illinois at Urbana-Champaign, Urbana, IL 61801, USA.

[2] Corresponding author; U.S. Army Research Laboratory, AMSRL-WM-BD (ATTN: Betsy Rice), Aberdeen Proving Ground, MD 21005-5066, USA; e-mail: betsyr@arl.mil

0010-4655/99/$ - see front matter Published by Elsevier Science B.V.
PII S0010-4655(98)00203-3

$$V_{\mathrm{Tot}} = \sum_{i=1}^{N-1} \sum_{j>i}^{N} V(r_{ij}) \,. \tag{1}$$

The evaluation of Eq. (1) and the gradients are usually the most computationally demanding steps in a simulation, even if the functional forms for $V(r_{ij})$ are extremely simple. Brute force evaluation of Eq. (1) requires the calculation of at least $N(N-1)/2$ internuclear distances. In a molecular dynamics simulation, each integration step often requires the evaluation of Eq. (1) and its gradients more than once depending on the integration scheme that is chosen [2]. It is clear that methods to reduce the computational burdens associated with numerous evaluations of Eq. (1) are required. The most obvious recent approaches are to modify the codes for scalable platforms. However, modifications of existing algorithms designed to reduce the computational burdens associated with evaluation of Eq. (1) can be made to increase the serial performance and exploit scalable architectures to achieve enhanced performance. In this work, we present a modification of existing algorithms that were developed to reduce unnecessary computations of the internuclear distances for atom pairs used in the evaluation of Eq. (1).

Common strategies to reduce the computational demands associated with Eq. (1) include the use of simple functions to describe the pair interaction potentials, and the assumption that the interaction between two particles is negligible beyond a certain cutoff distance r_{cut}. The assumption of a cutoff distance in the interaction potential allows for a reduction in computational time, since the interaction between atoms separated by distances exceeding r_{cut} do not need to be calculated. Unfortunately, the easiest and most direct way to determine the set of internuclear distances that are within r_{cut} is to evaluate all distances between all pairs, and eliminate those that exceed r_{cut}. This step requires a potentially large number of unnecessary calculations, and might be the most costly computational step in such a simulation.

The order N^2 method described in the preceding paragraph is due to the assumption of pair interaction potentials in Eq. (1). However, commonly used functions (such as Lennard-Jones or exp-6 forms) are too simple to correctly model all of the anisotropies that exist in many systems. Also, if chemical reactions in the condensed phase are being simulated, more sophisticated potential energy functions are required. Increasingly complex potential energy functions often use many of the internuclear distances evaluated for Eq. (1) more than once per evaluation of potential energy or force. An example is seen in the potential energy function used in the simulation of detonation [33]. In this example, the function that describes the interaction for all atoms in the system is

$$V_{\mathrm{Tot}} = \sum_{i=1}^{N} \sum_{j>i}^{N} \{ f_c(r_{ij}) [(2 - \overline{B}_{ij}) V_R(r_{ij}) - \overline{B}_{ij} V_A(r_{ij})] + V_{\mathrm{vdW}} \} \,, \tag{2}$$

where the first set of terms on the right-hand side of Eq. (2) (within the square brackets) contains the intramolecular interaction terms and includes many-body effects. The V_{vdW} term in Eq. (2) is a modified Lennard-Jones potential that describes the intermolecular interactions. The intra- and intermolecular interaction terms have different interaction ranges and thus sample different sets of internuclear distances out of the total set in the system. The many-body term in the intramolecular interaction portion of Eq. (2) has the form

$$\overline{B}_{ij} = \frac{1}{2} \left(\left\{ 1 + G \sum_{\substack{k=1 \\ k \neq i,j}}^{N} f_c(r_{ik}) \exp[m(r_{ij} - r_{ik})] \right\}^{-n} + \left\{ 1 + G \sum_{\substack{k=1 \\ k \neq i,j}}^{N} f_c(r_{jk}) \exp[m(r_{ij} - r_{jk})] \right\}^{-n} \right). \tag{3}$$

Evaluation of this term for a single i–j atom pair in Eq. (1) requires knowledge about the remaining $(N-2)$ internuclear distances. If a brute force calculation of the entire set of internuclear distances is performed for each evaluation of the intramolecular interaction between all atom pairs during evaluation of Eq. (1) using a potential of the form of Eq. (2), this simulation becomes order N^3.

A reduction of unnecessary calculations of internuclear distances can be accomplished through the use of the Verlet neighbor list [4]. This method requires the construction of a list of neighbors for each atom. An atom's neighbors are usually defined to be all of

the atoms that are within a distance slightly greater than the range of the interaction potential. Information about the neighbors is stored in arrays. For the duration of the simulation or until the lists are updated, each atom is assumed to interact only with the atoms on its neighbor list. The internuclear distances, interaction potentials, and forces are evaluated for each atom and its neighbors only. The list may be periodically updated to allow for the movement of atoms into or out of the interaction range. Brute force construction or update of the list requires the evaluation of all $N(N-1)/2$ internuclear distances. The method has been shown to be efficient when the system contains a relatively small number of atoms [2,5]. However, as the system becomes larger, the memory requirements for maintaining the neighbor lists become prohibitive. Also, as the mobility of the atoms becomes greater, either the frequency of lists updates must increase or the cutoff distance used in the definition of the neighbors must increase. Either of these requirements increases the computational demands of the Verlet neighbor list method. The example of the detonation simulation is one such case in which the mass flow (moving at supersonic speeds) would require large neighbor cutoff distances and frequent neighbor-list updates [3].

Alternative methods for the efficient determination of the interacting neighbors for each atom include grid or cell approaches [2,6–8]. These approaches partition the simulation space into grids or cells, to which the atoms are assigned by virtue of their positions relative to the cells. Since each cell has an unchanging set of neighboring cells that contain the volume within the distance r_{cut} of that cell, an atom associated with one of the cells has as its neighbors those atoms assigned to the same or neighboring cells. The implementations of these methods usually assign the atoms to the cells at each integration step. However, the same considerations used for the frequency of updating the Verlet neighbor-lists are applicable here. There is some overhead associated with these methods, and they are preferable only for systems that contain more than 1000 atoms [2]. These methods substantially reduce the number of unnecessary internuclear distance calculations in evaluating Eq. (1), but do not completely eliminate unnecessary computations.

In this work, we report modifications to grid-cell methods to further reduce the number of internuclear distance calculations in systems containing larger numbers of atoms. The approach we present is a modification of the conventional method of cell-linked lists as described in detail by Allen and Tildesley [2]. The results show a dramatic decrease in CPU requirements and are amenable to parallelization.

Brugé and co-workers have already provided geometric and systolic parallelization schemes for conventional implementations of Verlet neighbor lists and conventional cell-linked lists [8–10]. These have shown significant decreases in computation times and we refer the readers to such information. Our intent here is to modify the algorithms to accelerate both serial and scalable performance. We will describe the modifications and demonstrate the performance on serial platforms in this work. Future work will focus on scalability and further modifications to enhance performance. We are confident that some of the scalable methods set forth by Brugé and co-workers [8–10] will be applicable to these algorithms.

2. Method of cell-linked lists

2.1. Conventional method

The conventional method of cell-linked lists is well described by Allen and Tildesley [2]. We, like they, will describe our variation of the method in two dimensions, but the method can be generalized to include three dimensions. The modification we present is similar to one suggested by Allen and Tildesley: that the cell size be reduced so that no more than one atom can occupy a cell [2].

In the conventional method of cell-linked lists, the simulation space is partitioned into cells, the edges of which are no smaller than cutoff distance of the interaction potential. The atoms are then assigned to the various cells, by virtue of their position in the simulation space. A linked-list of the atom indices is created during the sorting procedure. Also, at the beginning of a simulation, an array that contains a list of cell neighbors for each cell is created. The list remains fixed unless the simulation space changes during the simulation (see, for example, Ref. [3]).

A cell i_{cell} has as its neighbors any cell that contains at least one point that is within the distance r_{cut} of any point within i_{cell}. Since the conventional method requires that the edges of each cell be no smaller

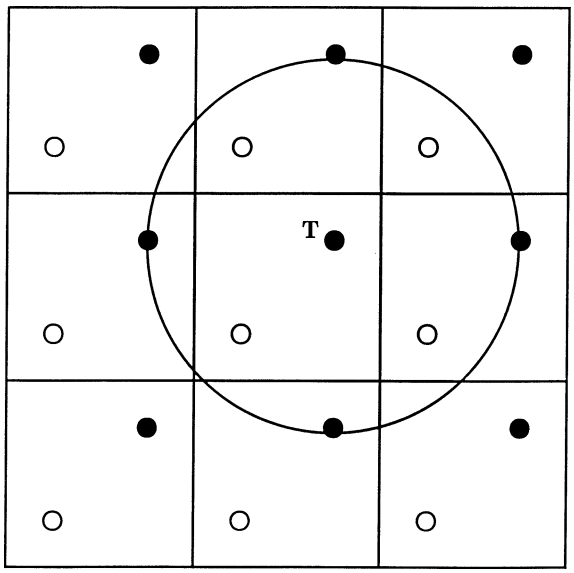

Fig. 1. Illustration of the conventional cell method in two dimensions. The simulation box is partitioned into 3×3 square cells. The edge length of each cell is r_{cut}. The shaded circle centered on atom T has radius r_{cut} and denotes the range of interaction for atom T. In this method, the eight outer cells are considered neighbors of the central cell that contains atom T.

than r_{cut}, each cell has eight nearest neighbors (we are assuming periodic boundary conditions in both dimensions of our two-dimensional example). These requirements ensure that all atoms that are within the interaction range of any atom within i_{cell} are assigned to the eight nearest-neighbor cells of i_{cell} or i_{cell} itself. All atoms occupying cells other than these are outside the interaction range of any atom located within i_{cell}. Fig. 1 illustrates the division of a region of the simulation space into cells. In this figure, both the x and y cell edges (denoted as l_x and l_y hereafter) equal r_{cut}. Evaluation of Eq. (1) occurs through looping over the cells using the linked-list of atoms rather than accessing the atom indices sequentially as written in Eq. (1).

This method dramatically reduces the number of unnecessary internuclear distance calculations that would result from a brute force calculation of all $N(N-1)/2$ internuclear distances. However, modifications can be made to further reduce the number of unnecessary distance calculations. In the conventional method, the distances between all atom pairs located within the rectangular area of $9l_xl_y$ are calculated. Assuming the limiting case $l_x = l_y = r_{\text{cut}}$, the area within which all distances are calculated is $9{r_{\text{cut}}}^2$. The area within the cutoff radius for a single atom is only $\pi{r_{\text{cut}}}^2$. Thus, the traditional cell-linked list method calculates distances between all atom pairs within an area that is almost three times larger (or more, since $l_i \geq r_{\text{cut}}$, $i = x$ or y) than that actually required for an atom. This dramatic difference is illustrated by comparing the area within the shaded circle centered on the atom labeled "T" with the area for the cell containing T and its neighboring cells in Fig. 11. The shaded circular area illustrates the range of interaction for atom T. Implementation of the conventional cell-linked list for this example would result in nine unnecessary internuclear distance calculations.

2.2. Modified method of cell-linked lists

The main modification of the method is in the definition of the sizes of the cells. By dividing the simulation space into smaller rectangular cells, each atom is surrounded by a group of cells that better approximates the area of interaction for that atom. For example, in Fig. 2, we have divided the original rectangular cells from Fig. 1 into fourths. The length of each cell is now $\frac{1}{2}r_{\text{cut}}$. The neighboring cells to that containing the labeled atom T are the surrounding first and second shells of cells. The area for this set of neighboring cells is $6.25\ {r_{\text{cut}}}^2$, which is approximately one-third smaller than the rectangular area that would be considered in the conventional approach (see Fig. 1). Also, the number of unnecessary internuclear distance calculations has been reduced to four. However, the number of neighboring cells to that containing atom T has increased from 8 to 24. Thus, the area has been substantially reduced, but the number of neighboring cells has increased by a factor of three. There is an increase in memory requirements associated with the linked-lists upon increasing the number of neighboring cells.

In the simple examples shown in Figs. 1 and 2, the division of the original cell size into fourths has reduced the number of unnecessary internuclear distance calculations from nine to four and a further reduction in cell size would probably not result in additional savings. A system whose atoms are arranged such that the density is not uniform might benefit from further reduction of the cell sizes to the point that the sphere of interaction of an atom is closely approximated by a set

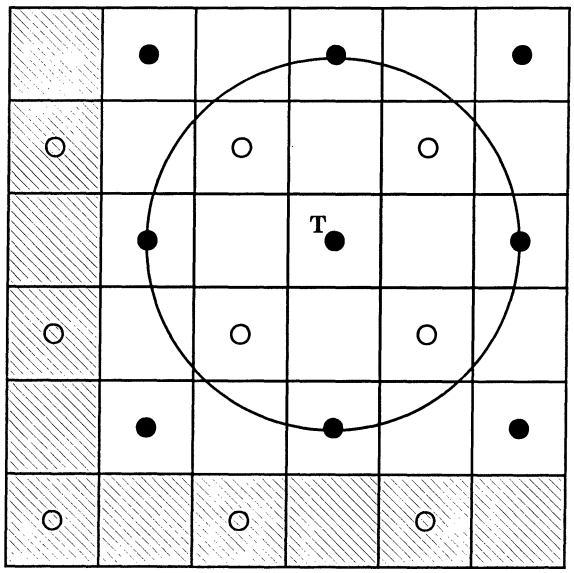

Fig. 2. Same as Fig. 1, except the simulation box is partitioned into 6×6 square cells. Edge length of each cell is $\frac{1}{2} r_{\text{cut}}$. The hatched area denotes the cells that are not considered to be neighbors of the cell containing atom T in the modified cell-linked list method.

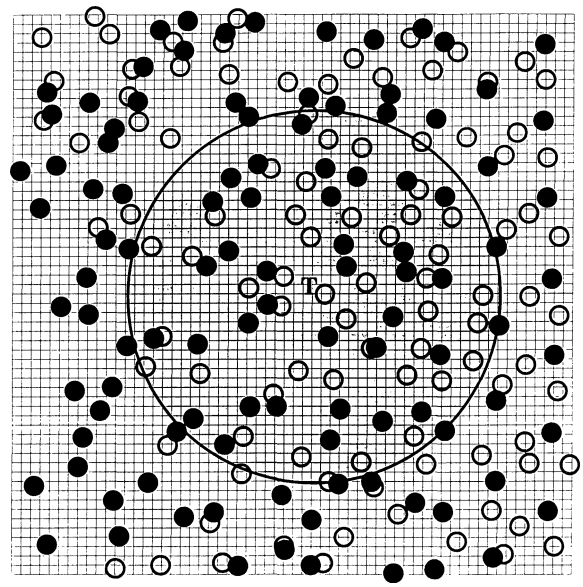

Fig. 3. Simulation Box partitioned into 60×60 square cells. The edge length of each cell is $1/20\ r_{\text{cut}}$. The atoms that are illustrated on this grid were taken from results of a molecular dynamics simulation of detonation, and correspond to the high dense region behind the detonation front (Ref. [3]).

of small rectangular cells. Such an example is given in Fig. 3, which has overlaid the positions of atoms behind a detonation wave [3] (a high dense region) onto a grid of cells with edge lengths $l_x = l_y = \frac{1}{20} r_{\text{cut}}$. It is clear that use of cells with the sizes shown in Figs. 1 or 2 would require many unnecessary internuclear distance calculations for the atomic arrangement in Fig. 3.

Note that many of the neighboring cells to that containing the labeled atom T in Fig. 3 are empty. There is overhead associated with determining whether a cell is occupied. Also, more memory is required to maintain the cell-linked list and the neighbor list, since as the cell sizes decrease the number of cells and the number of neighboring cells for each cell increase. While this method reduces the unnecessary distance calculations, there is a point at which the reduction in the size of the cell requires more computation in overhead than it saves in eliminating unnecessary distance calculations. The optimum cell size might vary from machine to machine and implementation to implementation. Therefore, it is desirable to use an algorithm that allows for the cell size to be changed easily to accommodate portability.

2.3. Off-set mapping method

As cell sizes decrease, memory requirements for storage of neighbor information increase and are potentially a limitation on the use of the modified cell-linked list scheme. This problem can be reduced by the determination of neighboring cells through a list of relative cell index offsets, similar in spirit to that presented in the Monotonic Logical Grid (MLG) approach [6,7]. After partitioning the simulation space into cells, each cell is assigned a grid cell index (i, j, k) that corresponds to its location in a Cartesian reference frame (x, y, z). Fig. 4 illustrates the two-dimensional grid overlaid on the simulation box shown in Fig. 2. In this example, the grid indices are assigned relative to cell (1,1), located at the lowermost cell on the left-hand side of the figure. The cell that contains the labeled atom T is located at the 4th column (x-direction) and the 4th row (y-direction). Thus, the grid index for this cell is (4,4). The set of cells that are within the interaction range (r_{cut}) for all points in cell (4,4) consists of the first and second nearest neighbors, each of which has a set of grid indices that can be described as relative offsets to (4,4). Each cell

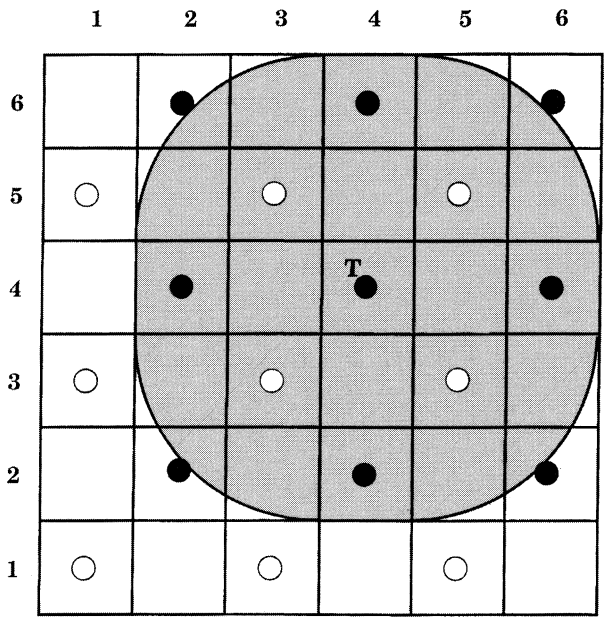

Fig. 4. Same as Fig. 2, with the "shaped" neighbor region (shaded area) illustrated.

in the simulation box has the same set of relative grid index offsets as (4,4). This set can be stored in a relative offset array, which is illustrated for this example in the upper portion of Fig. 5.

Determination of the relative cell index offsets of the neighbors is straightforward, particularly if the area encompassing the neighbors is rectangular. In this example, the rectangular area containing all neighboring cells has dimensions of $2r_{\mathrm{cut}} + l_i$, $i = x$ or y. However, the shape of the area containing the neighboring cells is not limited to a rectangle. Further reductions in unnecessary distance calculations can result if the area containing the neighboring cells resembles a circle. Since the set of neighbors must contain all of the area within the interaction range of any point within the cell, we want the minimum set of cells that make up this "neighbor region". Rounding the corners of the rectangular neighbor region will shape the neighbor region to approximate a circle. Again, using our simple example, we illustrate this in the shaded portion of Fig. 4. The rounded corners represent the portions of circles with radius r_{cut} that are centered on the corners of the cell that contains T. In this example, the number of cells containing the neighbor region is the same as that of the rectangular area. However, as the cell sizes

-2,2	-1,2	0,2	1,2	2,2
-2,1	-1,1	0,1	1,1	2,1
-2,0	-1,0	0,0	1,0	2,0
-2,-1	-1,-1	0,-1	1,-1	2,-1
-2,-2	-1,-2	0,-2	1,-2	2,-2

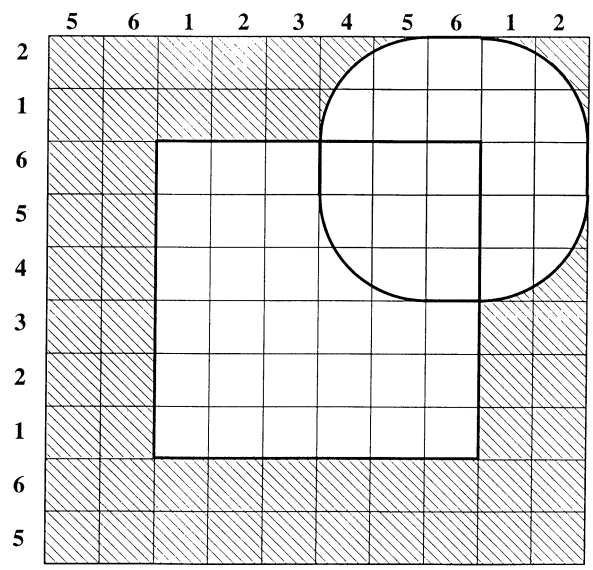

Fig. 5. Top: geometrically represents the offset list, with the relative offset numbers. Bottom: the same simulation box as in Fig. 4, surrounded by "ghost" cells (hatched area). This is the geometric representation of the mapping array. The numbers along the left-hand side and top of the figure indicate the packing of the mapping array.

are reduced, the number of cells containing the neighbor region will be less than those of the corresponding rectangular area, and the set of cells contained in the neighbor range will more closely approximate a circle.

To determine the minimum number of cells contained in the neighbor region, we first assume a rectangular simulation box that is larger than twice the cutoff radius in all dimensions. The box is then partitioned into cells of a desired size. At this point, assume that the central cell in this box has the grid index (0,0). Only the neighbor cells contained in one quadrant of this simulation box need to be identified, since the remaining neighboring cells that occupy the other quadrants can be generated from symmetry. We

will consider the top-right-hand quadrant in our illustration.

The process of identifying the neighbor cells in this quadrant begins with the calculation of the range of the cells along the x-axis. The grid index for the furthermost neighbor cell in the x-direction, xlen, is defined as xlen = floor($r_{\mathrm{cut}}/l_x + 1$). Each cell (ix,0) that is between between cells (0,0) and (xlen,0) is a neighbor of (0,0). (Due to the symmetry of the system, all cells (jx,0) that include or are between (0,0) and (−xlen,0) are also neighbor cells.) We then determine the range in the y-direction as follows: For each cell (ix,0) including or between (0,0) to (xlen,0), the grid index for the furthermost neighbor cell in the y-direction from cell (ix,0) is defined as ylen(ix) = floor(sqrt($r_{\mathrm{cut}}{}^2 - ((\mathrm{ix} - 1) \times l_x)^2)/l_y +$ 1). All cells that include or are between (ix,0) to (ix,ylen(ix)) are added to the list of neighbor cells. The process insures that any cell whose lower left-hand corner is less than r_{cut} from the upper right corner of the cell (0,0) is a neighbor. A sample FORTRAN code for this process is given in Appendix A.

In a simulation, when a cell with grid index (i, j) is selected, the neighboring cells are identified by simply adding the relative cell index offsets that are determined at the beginning of the simulation to the cell grid index (see Fig. 5 for the simple example presented in this work). If only half of the neighbors are required in the calculations, only the offsets in the first and second quadrants should be used, except those from (−xlen,0) to (−1,0). In our example here, in which we have found the neighbors in the upper right-most quadrant, we may just add the offsets from (−ix,1) to (−ix, ylen(ix)) with ix = −1 to xlen.

This method as described up to this point is sufficient for determining neighbors for cells that are far enough from the edges of the simulation box such that none of the neighbors should be minimum images. However, for cells on or near the edge of the simulation box, the method fails. Again, we use our simple example described in Fig. 2. In Fig. 5, we have reproduced the simulation box of Fig. 2, and surrounded it with a shell of "ghost" cells (hatched area) which is two cells deep in both dimensions. Overlaying the offset list (upper portion of Fig. 5) on cell (6,6), which is the geometric equivalent of simply adding the offset list indices to (6,6), would result in identifying as neighboring cells those with grid indices ranging from (4,4) to (8,8). This is clearly wrong, because the cells with indices greater than 6 are not defined. To remedy this, a mapping array has been developed to correctly map the two-dimensional relative grid cell index offsets to the appropriate set of neighbor cells while properly taking into account the boundary conditions of the simulation. The mapping array for the simple example given in this paper (assuming periodic boundary conditions in both directions) is shown in the bottom portion of Fig. 5. It is constructed using the column and row designators that border the top and left-hand side of the two-dimensional array in the bottom portion of Fig. 5. We illustrate its use as follows. In this simple example, one of the neighbor cells has relative cell index offset (0,2). Adding the relative cell index offset (0,2) to cell (6,6) addresses mapping array element (6,8). According to the mapping scheme, the element (6,8) of the mapping array contains the grid cell index (6,2). Cell (6,2), which corresponds to relative cell index offset (0,2), is the appropriate neighbor for cell (6,6) according to the periodic boundary conditions established for this example. As for the neighbor list, this map array increases in size with decreasing cell size.

By combining the neighbor offset list with the mapping array, the computation of and memory used for storing the neighbor information is kept at a reasonable level, even for very small cell sizes. There are four major arrays associated with this method. These are the list, the overlay, the listhead (which contains the index of the particle that is used to address the element of the linked-list array), and the map arrays. The size of the list array always equals the number of atoms. The size of the overlay array is proportional to the interaction range divided by the volume of the cell. The size of the listhead array equals the number of cells, and the size of the mapping array equals the total number of cells and ghost cells. When the number of cells is larger than the number of atoms, then the listhead and mapping arrays require the most memory. However, this method becomes inefficient before the number of cells equals the number of atoms. Therefore, in any reasonable use of this method, the list array has the largest memory requirement.

2.4. Distance lists

As noted earlier, often more complex functions reuse information in the evaluation of Eq. (1), such as those systems that use potentials described in Eqs. (2), (3). This form of interaction potential requires that the internuclear distances be used many times in a single evaluation of the potential energy or forces. Recalculations within a single integration step significantly increases the computer time required for a molecular dynamics simulation. To overcome this problem, we implemented lists that contain information about atom pairs for reuse in the evaluation of the interaction potential and forces. This would be unnecessary for models that assume pair-additive interaction potentials such as the Lennard-Jones or exp-6 potentials, since the internuclear distances for each pair are used only one time per evaluation of forces. But for functions such as those presented in Eqs. (2), (3), there are several terms that could benefit from storage of the internuclear distances. These include the $\exp(-g_{ik})$ terms, the $f(r_{ik})$ terms, and corresponding derivatives (x_{ik} and y_{ik}), distance r_{ik}, and the atom index of the neighbor. This information can be generated before or during every call to the potential energy and force subroutine using the linked-list method and neighbor list. If the distance is within the intramolecular interaction range, all information that can be reused is calculated and stored. Given an atom pair i–j, the stored information corresponding to that pair can easily be accessed during the evaluation of the potential energy and forces for that pair. Since the number of atom pairs can be large compared to the number of atoms, blocking techniques can be used for the storage of the atom pair information to minimize the memory required to store the atom pair information. In a blocking method, the atom pair information is calculated and stored for only a small number of cells in the simulation space at a time. The potential for these cells is calculated, and as the atom pair information is no longer needed, it is replaced by atom pair information for other nearby cells that will be used next.

3. Results

Although the description of the procedure given in the preceding section was given in two-dimensional terms, the method will be tested in a three-dimensional application. Six cubic simulation boxes that differ in size have been chosen to evaluate this methodology. The six simulation boxes consist of $27(3\times3\times3)$, $64(4\times4\times4)$, $125(5\times5\times5)$, $216(6\times6\times6)$, $343(7\times7\times7)$ and $512(8\times8\times8)$ cubic cells. Each cell has edge lengths just greater than r_{cut}. The different simulation boxes will be denoted hereafter as simulation box 3, 4, 5, 6, 7, and 8, respectively. All calculations were performed serially on an SGI Onyx with four 195 MHz R10000 processors with one and a half gigabytes of main memory and four megabytes of secondary cache per processor.

The CPU time used to evaluate the internuclear distances using this method as a function of system size and particle density is given in Table 1. For the evaluation using the cell-linked list methods, we report only the times for actual evaluation, and do not include any initialization. The initialization, which includes setting up the mapping array and the relative cell offset list, is relatively fast, and is only done once. The times reported are the averages for twenty separate evaluations of neighbors, and the timings include the construction of the linked lists for each evaluation. To check the method, all atom pairs were calculated and compared to those calculated through from the brute force method. In Table 1, the variable N_{div} denotes the number of divisions along an edge of the simulation box. For example, for simulation box 3, $N_{\mathrm{div}} = 3$ partitions each of the three edges of the box into three sections. The simulation box has a total of twenty-seven cells. This value of N_{div} corresponds to the conventional cell-linked list method. $N_{\mathrm{div}} = 0$ indicates that the cell-linked list method has not been used, and all $N(N-1)/2$ internuclear distances are calculated. The calculations for $N_{\mathrm{div}} = 0$ will be denoted as “brute force” calculations.

It has been established that the conventional cell-linked list method is superior to the brute-force approach for systems in which the dimensions are large compared to the cutoff radius of the potential [8]. We have seen the same result in this study. Table 1 gives the times for evaluation of the internuclear distances as a function of particle density and N_{div} for the six

Table 1
Time (ms) required to evaluate internuclear distances for systems of different sizes and particle densities

N_{div}	Atoms per cell											
	27		64		125		216		343		512	
	Time	% Red.[a]	Time	% Red.[a]	Time	% Red.[a]	Time	% Red.[a]	Time	% Red.[a]	Time	% Red.[a]
Simulation Box consisting of $3 \times 3 \times 3$ cubic cells												
0[b]	65	44.0	370	42.3	1445	41.6	4418	40.2	11133	40.4	24996	39.6
3[c]	116	0	641	0	2475	0	7386	0	18684	0	41406	0
6	71	38.8	403	37.1	1438	41.9	4401	40.4	10741	42.5	23920	42.2
9	82	29.3	336	47.6	1073	56.6	3473	53.0	8577	54.1	18608	55.1
12	98	15.5	407	36.5	1294	47.7	3324	55.0	7760	58.5	16626	59.8
15	146	−25.9	503	21.5	1406	43.2	3409	53.8	7485	59.9	15116	63.5
18	201	−73.3	615	4.1	1619	34.6	3877	47.5	7931	57.6	15863	61.7
21	258	−122.4	782	−22.0	1972	20.3	4451	39.7	9268	50.4	17833	56.9
Simulation Box consisting of $4 \times 4 \times 4$ cubic cells												
0[b]	339	−23.7	1926	−25.8	7368	−25.9	22238	−26.3	55979	−26.6	125085	−27.1
4[c]	274	0	1531	0	5853	0	17614	0	44220	0	98443	0
8	193	29.6	959	37.4	3504	40.1	10424	40.8	25934	41.4	57450	41.6
12	203	25.9	799	47.8	2807	52.0	8371	52.5	20405	53.9	44077	55.2
16	271	1.1	1064	30.5	3061	47.7	7869	55.3	18126	59.0	40565	58.8
20	364	−32.8	1222	20.2	3475	40.6	8135	53.8	17743	59.9	36886	62.5
24	489	−78.5	1454	5.0	3890	33.5	9500	46.1	19406	56.1	38539	60.9
28	667	−143.4	1914	−25.0	4704	19.6	11194	36.4	23138	47.7	44388	54.9
Simulation Box consisting of $5 \times 5 \times 5$ cubic cells												
0[b]	1248	−136.4	7056	−137.0	27028	−136.2	80968	−137.0	204402	−137.0	457627	−137.0
5[c]	528	0	2977	0	11442	0	34159	0	86245	0	193099	0
10	374	29.2	1874	37.1	6865	40.0	20249	40.7	50833	41.1	112994	41.5
15	397	24.8	1735	41.7	5627	50.8	16318	52.2	39879	53.8	86837	55.0
20	552	−4.5	2102	29.4	6000	47.6	15496	54.6	35812	58.5	78217	59.5
25	719	−36.2	2382	20.0	6768	40.8	16008	53.1	35042	59.4	71033	63.2
30	964	−82.6	2907	2.4	7858	31.3	18657	45.4	40285	53.3	78124	59.5
35	1346	−154.9	3798	−27.6	9670	15.5	21814	36.1	46529	46.1	84956	56.0
Simulation Box consisting of $6 \times 6 \times 6$ cubic cells												
0[b]	3688	−299.1	20721	−304.6	79091	−302.4	237039	−299.0	601276	−303.9	1355015	−305.0
6[c]	924	0	5121	0	19654	0	59413	0	148883	0	334566	0
12	663	28.2	3198	37.6	11988	39.0	35114	40.9	87566	41.2	195512	41.6
18	771	16.6	2991	41.6	9928	49.5	28189	52.6	68481	54.0	150101	55.1
24	973	−5.3	3618	29.3	10518	46.5	26943	54.7	62025	58.3	135247	59.6
30	1278	−38.3	4080	20.3	11851	39.7	27760	53.3	60281	59.5	123254	63.2
36	1683	−82.1	5214	−1.8	13577	30.9	34181	42.5	69848	53.1	136088	59.3
42	2347	−154.0	6876	−34.3	16745	14.8	38511	35.2	80343	46.0	148248	55.7
Simulation Box consisting of $7 \times 7 \times 7$ cubic cells												
0[b]	9174	−527.9	51667	−537.5	197044	−531.1	594673	−538.8	1515307	−542.5	3395131	−540.3
7[c]	1461	0	8105	0	31224	0	93086	0	235849	0	530254	0
14	1055	27.8	5032	37.9	19108	38.8	55661	40.2	139306	40.9	310958	17.0
21	1221	16.4	4787	40.9	15816	49.3	44021	52.7	108887	53.8	238017	51.3
28	1548	−6.0	5770	28.8	16736	46.4	42627	54.2	98612	58.2	210469	62.7

Table 1 — continued

N_{div}	Atoms per cell											
	27		64		125		216		343		512	
	Time	% Red.[a]	Time	% Red.[a]	Time	% Red.[a]	Time	% Red.[a]	Time	% Red.[a]	Time	% Red.[a]
35	2017	−38.1	6544	19.3	18880	39.3	43812	52.9	97988	58.5	200075	67.0
42	2665	−82.4	8243	−1.7	22687	27.3	54302	41.7	110560	53.1	213922	68.7
49	3740	−156.0	10937	−34.9	27262	12.7	60889	34.6	127843	45.8	234908	66.5
Simulation Box consisting of $8 \times 8 \times 8$ cubic cells												
0[b]	20360	−840.4	114378	−823.6	438915	−841.9	1329823	−848.9	3403575	−866.0	7888051	−905.7
8[c]	2165	0	12384	0	46599	0	140151	0	352334	0	784365	0
16	1570	27.5	8025	35.2	28340	39.2	82872	40.9	208860	40.7	456941	41.7
24	1819	16.0	7494	39.5	23296	50.0	66534	52.5	163040	53.7	355385	54.7
32	2312	−6.8	9027	27.1	24763	46.9	63588	54.6	147941	58.0	322611	58.9
40	3010	−39.0	9976	19.4	28035	39.8	65405	53.3	146368	58.5	303279	61.3
48	4107	−89.7	12579	−1.6	33962	27.1	81331	42.0	165208	53.1	326885	58.3
56	5861	−170.7	16931	−36.7	40796	12.5	91196	34.9	191219	45.7	358214	54.3

[a] Percent reduction of execution time relative to that using the conventional cell-linked list method.
[b] Brute Force method (see text).
[c] Conventional cell-linked list method (see text).

Table 2
Number of unnecessary internuclear distance calculations for various system sizes and particle number densities

N_{div}	No. of atoms per cell					
	27	64	125	216	343	512
Simulation Box consisting of $3 \times 3 \times 3$ cubic cells						
Req. [a]	26432	174460	684208	2257896	5646287	12940352
0[b]	238924	1317668	5009417	14745300	37232143	82604224
3[c]	238924	1317668	5009417	14745300	37232143	82604224
6	110516	688676	2479917	7580688	18662645	42348736
9	75223	411940	1449075	4994196	12701118	27410252
12	49109	316868	1388325	3800304	9516071	21179584
15	45128	252972	968867	2834820	7208556	15228564
18	40100	203688	764841	2169240	5614910	13097784
21	37120	176216	683984	1848988	4804972	11715904
Simulation Box consisting of $4 \times 4 \times 4$ cubic cells						
Req[a]	70304	436860	1696928	5594028	13769740	31587760
0[b]	1421824	7949700	30299072	89950548	227164436	505266768
4[c]	558688	3100036	11799072	34709844	87868020	194888272
8	293332	1609092	6111572	17727060	45043784	99467856
12	170464	948712	3858284	11596116	29810540	63704884
16	132992	825732	3224804	8838660	21984672	49283664
20	115276	603780	2315872	6565260	16714388	35911728
24	93200	463576	1770764	5227860	13790044	30060984
28	82340	427036	1533444	4626168	12218292	27882672

Table 2 — continued

N_{div}	No. of atoms per cell					
	27	64	125	216	343	512
Simulation Box consisting of $5 \times 5 \times 5$ cubic cells						
Req. [a]	145428	881372	3404448	11180808	27490362	62783136
0[b]	5548197	31114628	118658052	353305692	891621013	1985184864
5[c]	1083072	6026628	22954927	67537692	171020888	379552864
10	574333	3114628	11919888	34368192	87661399	193184864
15	324597	2122036	7765753	22393692	57732571	125439420
20	267238	1584228	6192889	17090928	42125062	95162464
25	222038	1121124	4431177	12592412	32066215	69197976
30	175870	922576	3583698	9951192	29202191	63713352
35	168950	857812	3317824	8815208	24289163	53251408
Simulation Box consisting of $6 \times 6 \times 63$ cubic cells						
Req. [a]	259752	1555996	5988268	19614300	48193916	109755152
0[b]	16743444	93988580	358498232	1068753540	2696284912	6005484784
6[c]	1863096	10381028	39560732	116411268	294833524	654601456
12	995060	5349092	20576920	59094372	151113168	332557552
18	644208	3646756	13406952	38402436	99192204	216103172
24	476948	2704100	10576944	29333220	73346640	163167472
30	391924	1870044	7549532	21371448	54740844	118474024
36	299248	1687464	6011796	18939108	50250076	108856920
42	282396	1579524	5666484	15456412	41272636	90945600
Simulation Box consisting of $7 \times 7 \times 7$ cubic cells						
Req. [a]	427310	2508732	9629888	31381752	77230911	175732480
0[b]	42451120	238425444	909481487	2713097076	6843353865	15244669440
7[c]	2943694	16446820	62700237	184621812	467483959	1038037504
14	1577899	8456292	32650934	93604704	239630154	526643712
21	1008145	5767564	21274439	60746676	156949686	342409612
28	753414	4255620	16652561	46450464	116162470	257650176
35	612874	3008764	11865187	33597004	90128832	196734024
42	460190	2671232	10396442	29836104	78872346	171449640
49	435242	2468668	9364454	24132408	64829056	143171928
Simulation Box consisting of $8 \times 8 \times 8$ cubic cells						
Req. [a]	639300	3787580	14510808	47190180	116163092	263943792
0[b]	94905276	533066948	2033457192	6068049756	15304238828	34095663504
8[c]	4392636	24507588	93457192	275240796	696938988	1547864464
16	2368072	12580036	48710196	139378524	357232184	784501136
24	1503420	8582740	31738324	90330972	233554028	510266172
32	1134968	6309060	24695332	69107820	172954384	382962064
40	919568	4477816	17572392	49688288	133800800	292228740
48	693232	3979096	15616800	44186460	116607292	254340984
56	695396	3785284	13906096	35478860	95880684	211718836

[a] The number of internuclear distances that are within the cutoff distance and are required to be calculated in an evaluation of Eq. (1) and its derivatives.

[b] Corresponds to the brute-force evaluation of the $N(N-1)/2$ internuclear distances in a system of N particles.

[c] Corresponds to the conventional method of cell-linked lists.

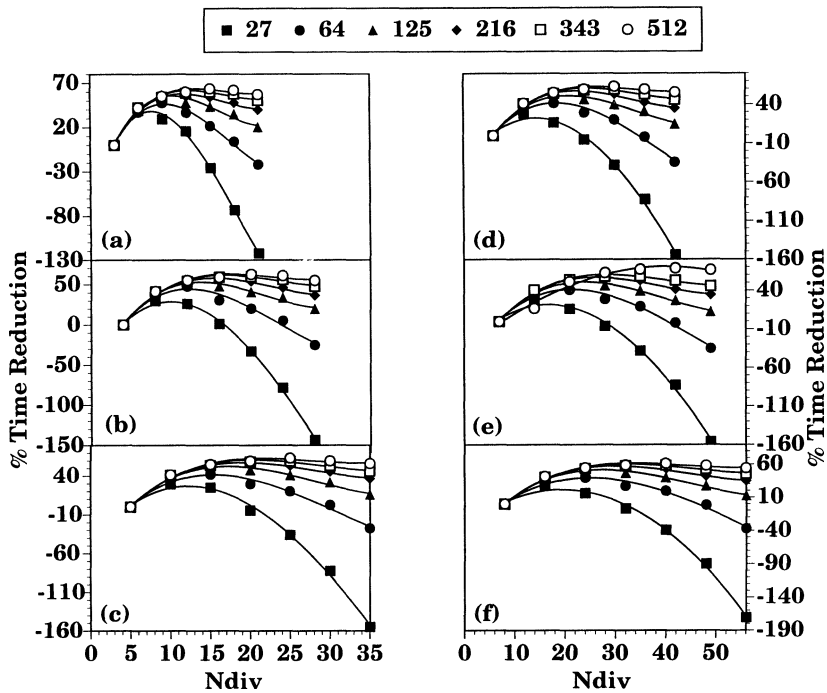

Fig. 6. Percent time reduction of the modified cell-linked list method over the conventional method as a function of the number of divisions (N_{div}) along each edge of the simulation box. The left-most point in each frame corresponds to the number of divisions required for the conventional cell-linked list method. (a) Simulation box 3 (27 unit cells); (b) simulation box 4 (64 unit cells); (c) simulation box 5 (125 unit cells); (d) simulation box 6 (216 unit cells); (e) simulation box 7 (343 unit cells); and (f) simulation box 8 (512 unit cells). Particle densities per cell are represented by symbols shown in the legend. Symbols are connected by a cubic polynomial fit, as a guide to the eye.

simulation boxes and the different methods. For simulation box 3, the execution times of the conventional and modified cell-linked list methods for low densities are greater than that of the brute force method. At higher densities, there is a slight speed-up using the modified cell-linked list method over the brute force approach. Note that for all densities for simulation box 3, the conventional method is slower than the brute force method. For systems that are larger than simulation box 3, however, the performance of the conventional and modified cell-linked list methods given here are superior to that of the brute force method. For the largest simulation box (box 8), there is a 90–97% reduction in CPU time over the brute force method.

Further comparison of the modified method will be made to the conventional cell-linked results rather than those using the brute-force method. Table 1 provides a percent reduction in time using the modified cell-linked list method over the conventional method, and Fig. 6 provides an illustration. Each curve in each frame of Fig. 6 shows that the percent time reduction first increases with increasing N_{div}, then decreases as N_{div} becomes larger. The subsequent decrease in performance with increasing values of N_{div} becomes more pronounced for systems with low particle densities. For example, the curves for densities of 27 and 64 particles per cell show that the modified cell-linked list method is much slower than the conventional method at large values of N_{div}. Conversely, the percent time reduction at large values of N_{div} for high densities (> 343 particles per cell) are only slightly less than the maximum value, indicating further time reduction does not necessarily occur with increased partitioning of the simulation space (reduction in cell size). This effect suggests that although the number of unnecessary internuclear distance calculations is decreasing with increasing N_{div} (see Table 2), the computational costs for the overhead associated with using a smaller cells is increasing and will eventually outweigh the savings realized from the reduced number of internu-

clear distance calculations.

4. Conclusions

It is clear that as advances in scalable architectures continue, more sophisticated molecular simulations will be attempted that require more atoms and more complex interaction potentials. It is because of this expectation that we have modified the traditional cell-linked list method to reduce unnecessary internuclear distance calculations for larger and more complex systems. We have shown a significant increase in speed of the evaluation of information needed for a molecular simulation through the reduction of unnecessary internuclear distance calculations. Although we have developed this algorithm for acceleration on serial machines, future efforts will invoke strategies for further increased performance on scalable architectures.

Acknowledgements

This work was supported under the DoD High Performance Computing Software Support Initiative (CHSSI) for Computational Chemistry and Material Science.

References

[1] D.L. Thompson, Modern Methods for Multidimensional Dynamics Computations in Chemistry (World Scientific, New Jersey, 1998).
[2] M.P. Allen, D.J. Tildesley, Computer Simulation of Liquids (Oxford University Press, New York, 1990).
[3] B.M. Rice, W. Mattson, J. Grosh, S.F. Trevino, Phys. Rev. E 53 (1996) 611.
[4] L. Verlet, Phys. Rev. 159 (1967) 98.
[5] J.J. Morales, L.F. Rull, S. Toxvaerd, Comput. Phys. Commun. 56 (1989) 129.
[6] J. Boris, J. Comput. Phys. 66 (1986) 1.
[7] S.G. Lambrakos, J.P. Boris, J. Comput. Phys. 73 (1987) 183.
[8] F. Brugé, J. Comput. Phys. 104 (1993) 263.
[9] F. Brugé, S.L. Fornili, Comput. Phys. Commun. 60 (1990) 31.
[10] F. Brugé, S.L. Fornili, Comput. Phys. Commun. 60 (1990) 39.

Appendix A

```
! Cutoffr is the cut off radius. Maxdim is the maximum coordinates for the simulations,
! mindim is the minimum. Ndiv is the number of divisions that the simulation is divided into.
! All of these are arrays of length 2.
cr2 = cutoffr * cutoffr
clen = (maxdim - mindim) / ndiv
len = int(cutoffr / clen) + 1
maxlen = len(2)

! Iterate from the cell immediately next to the test cell to the last cell in the x direction.
! Since the height above the test cell is always the same as the height above the cell immediately
! next to it we don't calculate it here. We start at 2 just for array index reasons.
do i = 2,len(2) + 1

 ! Calculate the height above the current cell.
 lengths(i) = floor(sqrt(cr2 - ((i - 2) * clen(2))**2)/clen(1) + 1)

enddo

! Taking advantage of the above mentioned symmetry
lengths(1) = lengths(2)
n = 0

! Time to replicate the cells for all quadrants and create the offset list.
! Loop over every cell along the x dimension.
do i = -len(2),len(2)

  ai = abs(i) + 1

  ! Loop over every cell along the y axis for column i
  do j = -lengths(ai),lengths(ai)

   ! Don't include cell (0,0)
   if(i .ne. 0 .or. j .ne. 0) then

    n =n + 1
    overlay(n,1) = i
    overlay(n,2) = j

   endif
  enddo
enddo
```