Theory Comput. Systems **37**, 295–318 (2004)
DOI: 10.1007/s00224-003-1071-0

Theory of Computing Systems
© 2004 Springer-Verlag New York Inc.

A Note on N-Body Computations with Cutoffs*

Marc Snir

Computer Science Department, University of Illinois at Urbana-Champaign,
Urbana, IL 61801, USA
snir@cs.uiuc.edu

Abstract. We provide a theoretical analysis of the communication requirements of parallel algorithms for molecular dynamic simulations. We describe two commonly used algorithms, space decomposition and force decomposition, and analyze their communication requirements; each is better in a distinct computation regime. We next introduce a new hybrid algorithm that further reduces communication. We show that the new algorithm is optimal, by providing a matching lower bound.

1. Introduction

1.1. *Outline of Paper*

We consider in this paper parallel algorithms for molecular dynamic simulations that use finite difference methods. Such simulations compute the movement of interacting atoms at discrete time intervals by repeatedly computing the forces acting on atoms and updating accordingly the position and velocity coordinates of these atoms. These simulations are very important in biomolecular research, as they allow us to understand the behavior of biological systems at the mesoscale level. Molecular simulations are expected to become increasingly important in drug design and in various applications of material sciences. However, these simulations are very time consuming and may require massive parallelism to be performed in acceptable time [2]. Several parallel implementations of the finite difference method have been analyzed in the past in some detail [11], [13]. The analysis has shown that communication requirements may be a significant impediment to the scalability of these algorithms.

* The work described in this paper was performed while the author was at IBM Research.

This paper presents a communication optimal algorithm for molecular dynamic simulations that improves on previous algorithms. We focus on the computation of intermolecular forces, which usually is the most time-consuming part of a molecular dynamic simulation. The next section summarizes the usual setting for such simulations and motivates the definition in Section 1.3 of the generic problem that we study in this paper, which we call “N-body computation with cutoff.” We proceed in Section 2 with an introduction of the computation model used in this paper. We present and analyze in Section 3 the two main algorithms that have been used in the past: space decomposition and force decomposition. We introduce in Section 4 a new algorithm that further reduces communication. In Section 5 we show that this new algorithm is optimal, by providing a matching lower bound on communication.

1.2. *Molecular Dynamics*

We consider in this paper N-body computations, as used in molecular dynamics. A molecular system of n atoms is simulated by repeatedly computing the forces acting between atoms and updating the atom coordinates accordingly.

The force applied on an atom is the vector sum of the pairwise interactions of that atom with all other atoms in the system. Those pairwise interactions can take several forms, as described in [1]. These include *intramolecular* forces that occur between atoms that are close-by in the same molecule and *intermolecular* forces that occur between any pair of atoms.

There are two types of intermolecular forces: *Lennard–Jones* forces and *Coulomb* forces. The magnitude of the Lennard–Jones force between two atoms a and b is given by a term of the form

$$U_{\mathrm{LJ}} = \frac{A_{ab}}{r_{ab}^{12}} - \frac{B_{ab}}{r_{ab}^{6}},$$

where r_{ab} is the distance between the two atoms and A_{ab} and B_{ab} are constants that depend on the atom types; $A_{ab} = A_{ba}$ and $B_{ab} = B_{ba}$. The magnitude of the Coulomb force is computed by a term of the form

$$U_{\mathrm{C}} = \frac{k q_a q_b}{r_{ab}},$$

where q_a and q_b are the atom charges and k is Coulomb’s constant. The Lennard–Jones forces decay rapidly with distance so that, usually, a *cutoff* distance is used, and forces are computed only for atoms that are within this cutoff. Coulomb forces, on the other hand, decay slowly, so that one needs to account for all pairwise interactions. However, it is common to simulate a system that consists of a periodic lattice (i.e., a rectangular cell that is replicated infinitely in all three dimensions). In such a case, one can use the *Ewald* method [7], [1] where the computation of the Coulomb forces is divided into two fast converging sums: a *real-space* part, where interactions are computed directly only for atoms within a cutoff distance, and a *k-space* part, where long-range interactions are computed in a transform space. By suitably balancing the two computations, one can reduce the asymptotic complexity to $O(n^{1.5})$. The use of a periodic system not only simplifies the computation of Coulomb forces; it also leads to a more realistic simulated system, as it avoids the "nonphysical" water–vacuum layer one has at the boundary of the simulated cell in an aperiodic system.

The asymptotic complexity of the Coulomb force computation can be further reduced to $O(n)$ by using a fast multipole algorithm [9], [8]. However, it is not clear that fast multipole algorithms are faster, in practice, for system sizes and machine sizes of interest [4].

1.3. *General Problem Formulation*

We focus in this paper on the more time-consuming part of molecular dynamic computations, namely the computation of intermolecular forces between all pairs of atoms within a cutoff distance. This includes the computation of Lennard–Jones forces and the real-space part of the Coulomb forces. We ignore the computation of the intramolecular forces and the k-space part of the Ewald computation.

We assume that the simulation uses a cubic cell $\mathbf{D}$ of dimensions $d \times d \times d$, replicated infinitely in all three dimensions. The results can be easily extended to rectangular cells of bounded aspect ratio, or to an aperiodic system. This cell contains a set $\mathcal{A} = \{a_1, \ldots, a_n\}$ of n atoms. We denote by $\mathbf{x}(a) = (x_1(a), x_2(a), x_3(a))$ the location of atom a. The atom location varies with time. We omit here and below the implicit time argument t from our notation.

We use

$$D(\mathbf{x}, \mathbf{y}) = \left(\sum_{i=1}^{3} (x_i - y_i)^2 \right)^{1/2}$$

for the Euclidian distance in $\mathbb{R}^3$. We define the *cyclic distance* between two points $\mathbf{x}$ and $\mathbf{y}$ to be

$$\tilde{D}_d(\mathbf{x}, \mathbf{y}) = \left(\sum_{i=1}^{3} ((x_i - y_i) \bmod d)^2 \right)^{1/2}$$

and define

$$\tilde{D}_d(a, b) = \tilde{D}_d(\mathbf{x}(a), \mathbf{x}(y)).$$

Let $near_a(b)$ be the copy of atom b that is nearest to atom a. Then $\tilde{D}_d(a, b) = D(a, near_a(b))$; $\tilde{D}_d(a, b)$ is the shortest Euclidian distance between a copy of atom a and a copy of atom b.

A cutoff distance c is used. We assume without loss of generality that $c < d/2$. This implies that, for any two atoms a and b, there is at most one copy of b that is within the cutoff distance from a.

This paper analyzes the following generic problem:

N-body Problem with Cutoff. Given n atoms $a_1, \ldots, a_n$ in a cubic cell $\mathbf{D}$ of dimension d, and a cutoff $c < d/2$, the algorithm maintains for each atom a a small tuple of fixed parameters $\mathbf{p}(a)$ and a small tuple of variable coordinates $\mathbf{x}(a), \mathbf{x}'(a), \ldots$. These include atom position $\mathbf{x}(a)$. The algorithm performs the following:

Force computation: for each pair of atoms (a, b) such that $\tilde{D}_d(a, b) < c$, compute the force

$$\mathbf{f}_{ba} = \mathbf{f}(\mathbf{p}(b), \mathbf{p}(a), \mathbf{x}(near_a(b)) - \mathbf{x}(a)).$$

Force reduction: for each atom a compute the force

$$\mathbf{f}_a = \sum_{\tilde{D}_d(b,a)<c} \mathbf{f}_{ba}.$$

Atom update: For each atom a apply an update to the atom coordinates

$$\mathbf{x}(a), \mathbf{x}'(a), \ldots = \mathbf{U}(\mathbf{x}(a), \mathbf{x}'(a), \ldots, \mathbf{f}_a).$$

We make the following assumptions:

- The functions $\mathbf{f}()$ and $\mathbf{U}()$ can be computed in (low) constant time.
- Newton's third law: $\mathbf{f}(\mathbf{p}, \mathbf{q}, \mathbf{x}) = -\mathbf{f}(\mathbf{q}, \mathbf{p}, -\mathbf{x})$. The force that atom a exercises on atom b has the same magnitude and a reverse direction than the force that atom b exercises on atom a.

An important feature of molecular dynamic simulations (as distinct from gravitational simulations) is that density is roughly constant. To simplify discussion, we make the following assumption:

Assumption 1 (Constant Density Assumption). *Let $\rho = n/d^3$ be the average density of the system. Then there exist constants δ and ε, such that any cube or ball of volume $V \geq \delta$ contains at least $\rho(1-\varepsilon)V$ and at most $\rho(1+\varepsilon)V$ atoms.*

We assume that $c^3 \gg \delta$, so that the system is homogeneous at the scale of the cutoff distance. With this assumption, then the number of interatomic forces acting on one atom is at most

$$(1+\varepsilon)\tfrac{4}{3}\pi c^3 \rho = (1+\varepsilon)\tfrac{4}{3}\pi (c/d)^3 n$$

and at least

$$(1-\varepsilon)\tfrac{4}{3}\pi c^3 \rho = (1-\varepsilon)\tfrac{4}{3}\pi (c/d)^3 n. \tag{1}$$

1.4. *Typical Parameters*

We shall analyze the complexity of the generic problem as a function of the following parameters:

- n, the number of atoms;
- c/d, the relative cutoff distance; and
- p, the number of processors.

Although the analysis is valid for any combination of values, it is worthwhile keeping in mind typical values for these parameters.

A typical molecular dynamic simulation may involve between $n = 32,000$ and $n = 200,000$ atoms [4]; as computers become more powerful, larger systems are simulated.

Cell size may vary between $d = 70$ Å for small systems and $d = 120$ Å for large systems. Cutoff distances in the range 7 Å $\leq c \leq$ 14 Å are typical, so that $\frac{1}{15} \leq c/d \leq \frac{1}{5}$.

Molecular dynamic simulations are routinely done on parallel systems with tens to hundreds of processors. The Blue Gene machine is expected to perform such simulations on up to 32,000 processors [2].

2. Programming Model and General Results

2.1. *Programming Model*

We are interested in parallel algorithms that solve the N-body computation with cutoff using p processors, with minimal computation and communication. We analyze the problem using the postal model [3]. In this model processors communicate via point-to-point messages. Those communications can be implemented by send and receive operations, or put/get operations, etc. The communication of a message of size k takes time $\ell + \tau k$; both sending and receiving processor are simultaneously busy for this time period; a processor can be involved in one communication at most at a time. Our upper bound results are expressed in this model. The lower bounds are on the *communication volume*, i.e., assume $\ell = 0$.

2.2. *Generic Parallel Algorithm for the N-Body Problem with Cutoff*

A parallel implementation of the N-body generic problem on p processors will allocate force computations and atom updates to the processors. Communications may be required to broadcast updated atom locations and to collect and sum forces computed at distinct nodes.

In the general formulation we use, a force may be redundantly computed at several nodes and the same atom coordinates may be redundantly updated at several nodes. While such a redundancy will increase computation, it could conceivably lead to reduced communication. Such a general algorithm is described by the following mappings:

$P(a)$ is the set of processors that update the coordinates of atom a. If the coordinates of atom a are updated at a unique processor, then we denote this processor by $p(a)$.

$P(a, b)$ is the set of processors that compute the force between atom a and atom b; $P(a, b) = P(b, a)$.

$B(i, a)$ is the set of processors to which processor i broadcasts the location of atom a. If a is held at a unique processor, then we omit the first argument i. $B(i, a) = \emptyset$ if $i \notin P(a)$, $i \in B(i, a)$, and $P(a, b) \subseteq \bigcup_i B(i, a)$.

$R(i, a)$ is the set of processors that contribute to the sum-reduction that yields the value of $\mathbf{f}_a$ at node $i \in P(a)$. If a is held at a unique processor, then we omit the first argument i. $R(i, a) = \emptyset$ if $i \notin P(a)$; if $i \in P(a)$ and $\tilde{D}_d(a, b) < c$, then $P(a, b) \cap R(i, a) \neq \emptyset$.

In addition, we denote by $A(i) = \{a : i \in P(a)\}$ the set of atoms that are updated by processor i, and by $F(i) = \{(a, b) : i \in P(a, b)\}$ the set of forces computed by processor i. $F = \bigcup_i F_i$ is the set of forces computed by the algorithm; $(a, b) \in F$ iff $\tilde{D}(a, b) < c$.

An algorithm for the solution of the N-body problem with cutoff consists of the following four phases:

Broadcast: For each atom a and each processor $i \in P(a)$, broadcast the location $\mathbf{x}(a)$ of atom a from processor i to each processor in the set $B(i, a)$.

Compute: For each pair $(a, b) \in F$, compute the force $\mathbf{f}_{ab}$ at each processor $i \in P(a, b)$.

Reduce: For each atom a and each processor $i \in P(a)$, sum-reduce all forces $\mathbf{f}_{ba}$ computed at nodes in $R(i, a)$ and collect the sum at processor i.

Update: For each atom a and each processor $i \in P(a)$, update the coordinates of a at i.

One can see that if the computation of the functions **f**() and **U**() are assumed to be “black-box” operations, and locations and forces are handled as atomic values, then this is the most general form of a parallel algorithm for the generic problem, with one qualification: We assume here that the set of processors that holds the coordinates of atom a at the end of the iteration is the same set that held the coordinates of a at the start of the iteration. One can further generalize by dropping this assumption. We suspect, but have no proof, that this last generalization cannot lead to further improvements.

An optimal distribution of atom updates and force computations will depend on the set of interacting atom pairs. As this set changes in time, the distribution has to be updated. However, the locations of the atoms change relatively slowly in molecular simulations. Thus, we assume that the distributions $P(a)$ and $P(a, b)$ and the communication patterns $B(i, a)$ and $R(i, a)$ may depend on atom locations, while ignoring the overhead for computing these distributions and remapping atoms.

3. Algorithms

Plimpton presents in [11] three parallel algorithms for molecular dynamics:

Atom Decomposition. Each processor is allocated n/p atoms. The processor computes the forces acting on these atoms and updates their coordinates.

Space Decomposition. Each processor is allocated a subvolume of the simulation cell. The processor handles the atoms in this subvolume.

Force Decomposition. The computed forces are evenly partitioned among processors.

All these algorithms perform the same computation; they are distinct in the distribution of computations to processors and in the communication pattern. We analyze below the last two of these algorithms and present next a new algorithm that combines the best features of these two algorithms.

3.1. *Space Decomposition*

We assume that p is perfect cube, to simplify notation. We denote each processor by a triplet (i, j, k), $0 \le i, j, k < p^{1/3}$. The simulation cell $\mathbf{D}$ is decomposed into $p = p^{1/3} \times p^{1/3} \times p^{1/3}$ cubic subcells $C(i, j, k)$, each of dimension $\tilde{d} = d/p^{1/3}$. The atoms in subcell $C(i, j, k)$ are allocated to processor (i, j, k). This processor computes all forces acting on atoms in its subcell and updates the atom coordinates. We have

$$p(a) = (\lfloor x_1(a)p^{1/3}/d\rfloor, \lfloor x_2(a)p^{1/3}/d\rfloor, \lfloor x_3(a)p^{1/3}/d\rfloor),$$

$$P(a, b) = \{p(a), p(b)\}.$$

Note that, in this formulation, each force is computed twice and no advantage is taken of symmetry.

We assume that $d^3/p > \delta$, so that each subcell contains at most $(1 + \varepsilon)n/p$ atoms. The algorithm distributes computation evenly: each processor computes at most $(1 + \varepsilon)^2 \frac{4}{3}\pi(c/d)^3 n^2/p$ forces.

We assume that the processors (= subcells) are logically organized in a cyclical three-dimensional grid. (This is a virtual grid—each processor can still directly communicate with any other.) The reduction phase of the algorithm requires no communication. Let $r = \lceil c/\tilde{d}\rceil = \lceil (c/d)p^{1/3}\rceil$. In the broadcast phase of the algorithm processor (i, j, k) communicate the positions of its atoms to all processors $(i + u, j + v, k + w)$, where $-r \le u, v, w \le r$, i.e., the broadcast covers a (cyclic) cube of dimension $2r+1$, centered at (i, j, k).

If $\tilde{D}_d(a, b) < c$, then $|(x_i(a) - x_i(b)) \bmod d| < c, i = 1, 2, 3$, so that $|(p_i(a) - p_i(b)) \bmod p^{1/3}| = |\lfloor x_i(a)/\tilde{d}\rfloor - \lfloor x_i(b)/\tilde{d}\rfloor \bmod p^{1/3}| \le \lceil c/\tilde{d}\rceil = r$. Thus, the broadcast phase covers all needed communications. The broadcast pattern is illustrated in Figure 1, for $r = 2$; for simplicity, we only illustrate a two-dimensional slice.

The broadcast can be implemented by a sequence of $6r$ circular shifts steps, where at each step data is shifted by +1 or -1 in one dimension. After the first $2r$ steps of positive

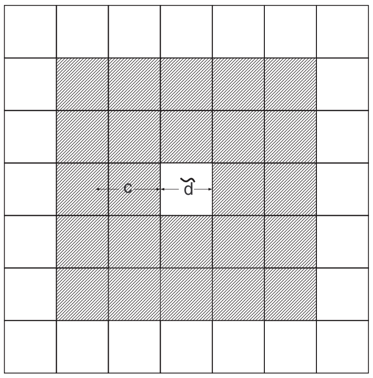

Fig. 1. Two-dimensional view of the communication pattern in space decomposition. Shaded cells exchange coordinates with the central cell.

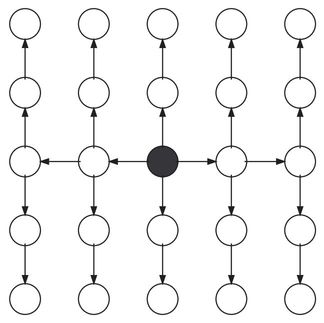

Fig. 2. First $4r$ steps of broadcast for the space decomposition algorithm.

and negative shifts in the first dimension, node (i, j, k) contains the coordinates of atoms for cells in the segment $(i+u, j, k)$, $-r \le u \le r$; after the next $2r$ steps of positive and negative shifts in the second dimension, node (i, j, k) contains the coordinates of atoms for cells in the square $(i+u, j+v, k)$, $-r \le u, v \le r$; the last $2r$ steps of shifts in the third dimension complete the broadcast. The first $4r$ steps of this broadcast algorithm are illustrated in Figure 2.

The communication complexity for this broadcast algorithm is bounded by

$$\begin{aligned} &6\ell r + \tau(1+\varepsilon)(2r + 2r(2r+1) + 2r(2r+1)^2)n/p \\ &\quad = 6\ell r + \tau(1+\varepsilon)((2r+1)^3 - 1)n/p. \end{aligned} \tag{2}$$

Note that this algorithm uses only nearest neighbor connections in the cyclical three-dimensional grid. A detailed pseudocode for the space decomposition algorithm appears in [12]. We also describe there several improvements to the basic algorithm. In particular, one can achieve better load balance when cells are small by pooling cells; and one can filter communications so as to ensure that each cell receives only the coordinates of atoms that interact with atoms within the receiving cell.

3.2. *Force Decomposition*

We present here a version of force decomposition similar to that used in [2].

Assume that p is a perfect square. We consider the processors to be organized into a $\sqrt{p} \times \sqrt{p}$ logical two-dimensional grid. Processors are labeled (i, j), $0 \le i, j < \sqrt{p}$. Atoms are allocated evenly to processors. Each processor (i, j) holds a set $A(i, j)$ of n/p atoms.

Let $A(i, *) = \bigcup_j A(i, j)$ and let $A(*, j) = \bigcup_i A(i, j)$. We allocate to processor (i, j) all forces due to interactions of atoms in $A(i, *)$ with atoms in $A(*, j)$. We thus have:

If $p(a) = (i, j)$, then $B(a) = \{(i, 0), \ldots, (i, \sqrt{p}-1), (0, j), \ldots (\sqrt{p}-1, j)\}$. That is, the positions of the atoms held at a processor (i, j) are broadcast to all processors in row i and all processors in column j. The broadcast requires

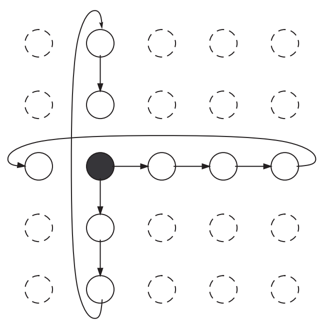

Fig. 3. Broadcast pattern for the force decomposition algorithm.

$2\sqrt{p}-2$ shifts, as illustrated in Figure 3. After this broadcast phase each processor (i, j) holds the positions of the atoms in $A(i,*) \cup A(*, j)$.

If $p(a) = (i, j)$, $p(b) = (i', j')$, $p(a) \neq p(b)$, and $\tilde{D}(a, b) < c$, then $P(a, b) = \{(i, j'), (i', j)\}$. The force $\mathbf{f}_{ba}$ is computed at the two processors that hold both the positions of a and of b, after the broadcast. This is illustrated in Figure 4.

Reductions can proceed either on columns or on rows. In the former case we have, if $p(a) = (i, j)$, then $R(a) = \{(0, j), \ldots, (\sqrt{p}-1, j)\}$.

We describe below in Section 3.3 a randomized allocation of atoms to processors that results, with overwhelming probability, into a balanced partition of forces to processors, so that each processor computes $\Theta((c/d)^3 n^2/p)$ interatomic forces.

The communication complexity for the broadcast phase is bounded by

$$2\ell\sqrt{p} + 2\tau\sqrt{p}(n/p) \;=\; 2\ell\sqrt{p} + 2\tau n/\sqrt{p}.$$

The reduce phase requires only communication on columns—the "reverse" of half of the broadcast algorithm. The communication complexity for the reduce phase is bounded by $\ell\sqrt{p} + \tau n/\sqrt{p}$ and the total communication complexity is bounded by

$$3\ell\sqrt{p} + 3\tau n/\sqrt{p}. \tag{3}$$

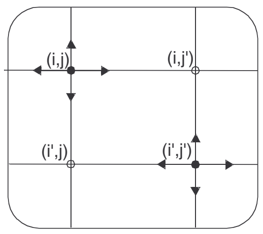

Fig. 4. Allocation of force computations in the force decomposition algorithm.

Our analysis has ignored the overhead of identifying, at each processor, the $\Theta((c/d)^3 n^2/p)$ pairs of atoms within the cutoff distance out of $(n/\sqrt{p}) \times (n/\sqrt{p}) = n^2/p$ pairs available at the processor after the broadcast phase. Testing each pair would require too much computation. However, it is possible to use faster algorithms. Given n points and a cutoff distance, one can find all m pairs within the cutoff distance in time $O(n \log n + m)$, rather than $O(n^2)$ [6]; given the constant density assumption, a simple $O(n+m)$ average complexity algorithm can be designed. Furthermore, as we assume that atom locations change slowly, it is possible to precompute the set of interacting pairs—using a slightly increased cutoff distance, so that the same precomputed structure can be used for many iterations. Therefore, we ignore the overhead of identifying interacting atom pairs.

3.3. *Probabilistic Analysis*

We show in this subsection that randomization can ensure, with overwhelming probability, that the force decomposition algorithm is load balanced. The proof uses Chernoff bounds for tails of binomial distributions. This assumes that the average number of forces per processor is sufficiently large. Specifically, we assume that

$$\frac{n}{\log_2 n} \geq \left(\frac{d}{c}\right)^3 \frac{k\sqrt{p}}{\frac{8}{3}\pi(1-\varepsilon)} \tag{4}$$

for some constant $k > \frac{3}{2}$. This assumption is easily satisfied for typical problem parameters.

We assume that atoms are partitioned deterministically across rows: each row i is allocated a set A_i of $n/\sqrt{p}$ atoms. The atoms are next randomly distributed to columns. For a fixed atom a and a fixed column j, the number $X(a, j)$ of atoms b that interact with a and have been allocated to column j is the sum of N_a independent Poisson trials, where $\frac{4}{3}\pi(1+\varepsilon)(c/d)^3 n \geq N_a \geq \frac{4}{3}\pi(1-\varepsilon)(c/d)^3 n$, and each trial has probability of success $1/\sqrt{p}$. The expected value is $\mu_a = N_a/\sqrt{p}$.

Let $\eta = 1/n^k$. Then, using (4), we have

$$\frac{\log_2(1/\eta)}{\mu_a} - 1 = \frac{k \log_2 n}{N_a/\sqrt{p}} - 1 \leq \frac{k\sqrt{p}\log_2 n}{\frac{4}{3}\pi(1-\varepsilon)(c/d)^3 n} - 1 \leq 1.$$

We now use the Chernoff bound (4.11) on page 72 in [10], to obtain that

$$\Pr(X(a, j) > 2\mu_a) \leq \eta.$$

If $X(a, j) \leq 2\mu_a$ for each atom a and each column j, then the number of forces computed at processor (i, j) is bounded by

$$\begin{aligned}\sum_{a \in A_i} X(a, j) &\leq \sum_{a \in A_i} 2N_a/\sqrt{p} \\ &\leq (n/\sqrt{p})2(\tfrac{4}{3})\pi(1+\varepsilon)(c/d)^3 n/\sqrt{p} \\ &= \tfrac{8}{3}\pi(1+\varepsilon)(c/d)^3 n^2/p.\end{aligned}$$

The probability of this occurring is at least

$$1 - n\sqrt{p}\eta \geq 1 - n^{3/2}\eta = 1 - n^{3/2-k}.$$

4. New Hybrid Method

If we compare the two algorithms described in the previous section, then we see that the space decomposition algorithm is preferable when p is small relative to d/c, whereas the force decomposition algorithm is preferable when p is large relative to d/c. In the extreme case where $p < (d/c)^3$, so that $d/p^{1/3} > c$, then each processor in the space decomposition algorithm communicates only with its 26 neighbors, whereas each processor in the force decomposition algorithm communicates with $2\sqrt{p} - 2$ other processors. At the other extreme, when $c = d$ and there is no cutoff, then, in the space decomposition algorithm, each processor communicates with all other $p - 1$ processors, while, in the force decomposition algorithm, it still is the case that each processor communicates with $2\sqrt{p} - 2$ other processors. The space decomposition algorithm takes advantage of locality and has less communication when the cutoff c is small; but the force decomposition algorithm avoids the expense of a direct broadcast to a large set of processors. The hybrid algorithm described in this section combines both ideas to minimize communication.

Consider a partition of atoms to p processors such that each processor holds $\Theta(n/p)$ atoms, and each atom is updated by a unique processor. We say that two processors p and q *interact* if each holds atoms $a \in A(p)$ and $b \in A(q)$ such that $\tilde{D}(a, b) < c$. Interacting processors communicate, either directly or indirectly, during an iteration. With space decomposition, each processor interacts with $O((c/d)^3 p)$ other processors. This is optimal, as each atom interacts with $\Omega((c/d)^3 n)$ other atoms, hence each processor interacts with

$$\Omega(((c/d)^3 n)/(n/p)) = \Omega((c/d)^3 p)$$

other processors. However, the broadcast scheme used in the algorithm, where each processor broadcasts its positions to all other processors it interacts with, is not optimal.

The broadcast scheme used by the force decomposition method has the property that for any two processors p_1 and p_2, there is a processor p_3 so that both p_1 and p_2 send all their data to p_3. Then p_3 can compute all interactions between atoms from p_1 and atoms from p_2. In general, one needs a communication scheme so that if processors p_1 and p_2 interact, then there is a processor p_3 so that both p_1 and p_2 send all their data to p_3. This reduces communication compared with a scheme where each pair of interacting processors need to exchange data. We show in this section that such a communication scheme can be defined, for space decomposition, so that each processor will broadcast its atom positions to only $O(\sqrt{(c/d)^3 p})$ other processors, i.e., to the square root of the number of processors it interacts with. This is superior to the simple space decomposition method and equal to the force decomposition method when $c = d$.

We use the same allocation of atoms to processors as described for the space decomposition method. The simulation cell is divided into $p^{1/3} \times p^{1/3} \times p^{1/3}$ subcells, each

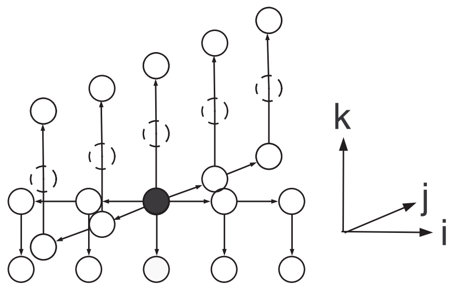

Fig. 5. Broadcast pattern for hybrid algorithm.

containing at most $(1+\varepsilon)n/p$ atoms. Each subcell is allocated to one processor. We assume that the processors are organized in a logical three-dimensional cyclical mesh.

Let $r = \lceil (c/d)p^{1/3} \rceil$. If processor (i, j, k) interacts with processors (i', j', k'), then $|(i-i') \bmod p^{1/3}| \le r$, $|(j-j') \bmod p^{1/3}| \le r$, and $|(k-k') \bmod p^{1/3}| \le r$. Let $\tilde{r} = \lceil \sqrt{r+1} \rceil$. If $0 \le m \le r$, then $m = m_2\tilde{r} + m_1$, where $0 \le m_1 \le \tilde{r} - 1$ and $0 \le m_2 \le \lfloor r/\tilde{r} \rfloor$.

The algorithm will have each processor (i, j, k) broadcast the positions of the atoms it holds to a set $B(i, j, k)$ of processors, where

$$\begin{aligned} B(i,j,k) = \{&(i+u, j, k-w_1),\ -r \le u \le r,\ 0 \le w_1 \le \tilde{r}-1\} \\ &\cup \{(i, j+v, k+w_2\tilde{r}),\ -r \le v \le r,\ 0 \le w_2 \le \lfloor r/\tilde{r} \rfloor\}. \end{aligned}$$

The communication pattern is illustrated in Figure 5, for $r = 2$ and $\tilde{r} = 2$.

Claim 1. *If processor (i, j, k) interacts with processor (i', j', k'), then there exists a processor $(\hat{i}, \hat{j}, \hat{k})$ such that $(\hat{i}, \hat{j}, \hat{k}) \in B(i, j, k) \cap B(i', j', k')$.*

Proof. Assume without loss of generality that $k = k' + w$, where $0 \le w \le r$ (all equalities are mod $p^{1/3}$). Then $k - k' = w = w_2\tilde{r} + w_1$, where $0 \le w_1 \le \tilde{r} - 1$ and $0 \le w_2 \le \lfloor r/\tilde{r} \rfloor$. Let $\hat{k} = k - w_1 = k' + w_2\tilde{r}$. Let $\hat{i} = i'$, and let $\hat{j} = j$. Then $(\hat{i}, \hat{j}, \hat{k}) = (i+u, j, k-w_1)$, where $|u| = |i - i'| \le r$ and $0 \le w_1 \le \tilde{r} - 1$; and $(\hat{i}, \hat{j}, \hat{k}) = (i', j'+v, k'+w_2\tilde{r})$, where $|v| = |j - j'| \le r$ and $0 \le w_1 \le \lfloor r/\tilde{r} \rfloor$. □

If $2r < p^{1/3}$, then the node $(\hat{i}, \hat{j}, \hat{k})$ is uniquely defined, and each force is computed at a unique node. As the construction is shift-invariant, each processor is allocated an equal number of cell pairs, so that the computation is load balanced.

We describe below the broadcast phase of this algorithm. We denote by $A := \mathit{Get}((u, v, w), B)$ the circular shift operation that assigns to variable A at each processor (i, j, k) the value of variable B at processor $(i+u, j+v, k+w)$, where all additions are modulo $p^{1/3}$.

Algorithm 1

```
let $A_0$ be the set of positions for atoms in a local cell
assert $|A_0| \leq (1+\varepsilon)n/p$
for $i := 1$ to $r$ do
    $A_i := Get((-1,0,0), A_{i-1})$
od
for $i := 1$ to $r$ do
    $A_{-i} := Get((1,0,0), A_{1-i})$
od
assert $A_{-r}, \ldots, A_r$ at processor $(i, j, k)$ holds all atom positions from processors $(i - u, j, k), -r \leq u \leq r$
let $B_0 = \langle A_{-r}, \ldots, A_r \rangle$
assert $|B_0| \leq (1+\varepsilon)(2r+1)n/p$
for $i := 1$ to $\tilde{r} - 1$ do
    $B_i := Get((0,0,1), B_{i-1})$
od
assert $B_0, \ldots, B_{\tilde{r}-1}$ at processor $(i, j, k)$ holds all atom positions from processors $(i - u, j, k + w), -r \leq u \leq r, 0 \leq w \leq \tilde{r} - 1$
for $i := 1$ to $r$ do
    $A_i := Get((0,-1,0), A_{i-1})$
od
for $i := 1$ to $r$ do
    $A_{-i} := Get((0,1,0), A_{1-i})$
od
assert $A_{-r}, \ldots, A_r$ at processor $(i, j, k)$ holds all atom positions from processors $(i, j - v, k), -r \leq v \leq r$
let $C_0 = \langle A_{-r}, \ldots, A_r \rangle$
assert $|C_0| \leq (1+\varepsilon)(2r+1)n/p$
for $i := 1$ to $\lfloor r/\tilde{r} \rfloor$ do
    $C_i := Get((0,0,-\tilde{r}), C_{i-1})$
od
assert $C_0, \ldots, C_{\lfloor r/\tilde{r} \rfloor}$ at processor $(i, j, k)$ holds all atom positions from processors $(i, j - v, k - w\tilde{r}), -r \leq v \leq r, 0 \leq w \leq \lfloor r/\tilde{r} \rfloor$
```

The communication complexity for this broadcast pattern is

$$\ell(4r + \tilde{r} + \lfloor r/\tilde{r} \rfloor - 1) + \tau(1+\varepsilon)(4r + (\tilde{r} + \lfloor r/\tilde{r} \rfloor - 1)(2r+1))n/p.$$

The reduction phase can be implemented be reversing the broadcast communication pattern. The total communication complexity is

$$2\ell(4r + \tilde{r} + \lfloor r/\tilde{r} \rfloor - 1) + 2\tau(1+\varepsilon)(4r + (\tilde{r} + \lfloor r/\tilde{r} \rfloor - 1)(2r+1))n/p.$$

One can show, by induction, that

$$\lceil \sqrt{r+1} \rceil + \lfloor r/\lceil \sqrt{r+1} \rceil \rfloor = \lfloor \sqrt{4r+1} \rfloor.$$

Thus, the communication complexity is

$$2\ell(4r + \lfloor\sqrt{4r+1}\rfloor - 1) + 2\tau(1+\varepsilon)(4r + (\lfloor\sqrt{4r+1}\rfloor - 1)(2r+1))n/p, \quad (5)$$

where $r = \lceil(c/d)p^{1/3}\rceil$.

There are two cases of interest:

Low parallelism, where the cutoff distance is smaller than the subcell size: $c < \tilde{d} = d/p^{1/3}$, or $p < (d/c)^3$. In a typical problem, where $d = 70$ Å and $c = 7$ Å, this means $p < 1000$. This encompasses the large majority of current runs. In the low parallelism case ($p < (c/d)^3$) each cell interacts only with its 26 neighbors. We have $r = 1$ and the communication complexity of the hybrid algorithm is

$$10\ell + 14\tau(1+\varepsilon)n/p. \quad (6)$$

In this case the hybrid algorithm generates as much communication, within a constant factor, as the space decomposition algorithm (equation (2)).

High parallelism, where the cutoff distance is large, relative to the cell size: $c \geq d/p^{1/3}$, or $p \geq (d/c)^3$. In such a case there are "long-distance" interactions between nonadjacent cells. This is relevant for massively parallel systems such as Blue Gene or for simulations that use a large cutoff distance.

In the high parallelism case ($p \geq (d/c)^3$), the communication complexity of the hybrid algorithm can be estimated as

$$8\ell(c/d)p^{1/3} \; + \; 8\tau(1+\varepsilon)(c/d)^{3/2}n/p^{1/2} + \text{lower-order terms.} \quad (7)$$

In the extreme case where $c = d$ (no cutoff) then we have the same communication volume, within a constant factor, as for the force decomposition algorithm (equation (3)).

In the low parallelism case the communication can be further reduced by refraining from transmitting the coordinates of atoms that are at least c apart from the cell boundaries; such atoms do not interact with atoms in other cells. This is illustrated, in two dimensions, in Figure 6. The volume of the "shell" of points at most c apart from the cell boundary is $\tilde{d}^3 - \max(0, (\tilde{d} - 2c)^3)$; this volume contains at most

$$\begin{aligned}&(1+\varepsilon)n(\tilde{d}^3 - \max(0, (\tilde{d} - 2c)^3))/d^3\\ &\quad = (1+\varepsilon)(1 - \max(0, (1 - 2p^{1/3}(c/d))^3))n/p\end{aligned}$$

points. The communication complexity of the hybrid algorithm in the low parallelism case becomes

$$10\ell + 14\tau(1+\varepsilon)(1 - \max(0, (1 - 2p^{1/3}(c/d))^3))n/p. \quad (8)$$

The communication volume is $\Theta((c/d)n/p^{2/3})$.

A practical implementation of the hybrid algorithm described in this section needs to address in detail several issues that are ignored in our analysis. This includes, in particular:

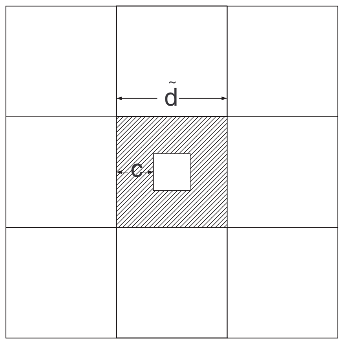

Fig. 6. Filtered communication. Only coordinates of atoms in shaded area are communicated by the cell in the center.

Atom Pair Filtering. Each processor needs to compute forces only between pairs of atoms that are less than c apart; this is a proper subset of the set of pairs of atoms received in the broadcast phase. The filtering problem can be handled as for the force decomposition algorithm, using precomputed lists and/or efficient computational geometry algorithms for finding pairs of points at distance $< c$.

Computing Intramolecular Forces. A benefit of the proposed hybrid algorithm is that it uses a volume decomposition of atoms, so that the two, three, and four body intramolecular force terms involve atoms that are within the same cell or in adjacent cells. Adjacent cells can exchange information on atom locations for bonds that cross cell boundaries, and intramolecular forces that act on an atom can be computed by the processor that "owns" that atom.

5. Lower Bounds

We prove in this section lower bounds on communication that show that the algorithm described in the previous section is optimal, among algorithms that perform minimal work. To simplify the (fairly tedious) derivations of this section we do not attempt to derive the best possible constants, or use the weakest preconditions.

Consider an instance of the N-body problem with cutoffs to be solved with p processors. We assume that $n \geq p$.

We use the notation introduced in Sections 1.3 and 2.2, with the following additions:

$n_i = |A(i)|$ is the number of atoms updated by processor i.
$f_i = |F(i)|$ is the number of forces computed by processor i.
m_i is the communication volume at processor i, i.e., the number of atom positions sent or received plus the number of forces sent or received by processor i.

The following inequalities are satisfied:

$$\sum_i n_i \geq n, \tag{9}$$

$$\sum_i f_i \geq (4/3)\pi(1-\varepsilon)(c/d)^3 n^2, \tag{10}$$

$$f_i \leq (n_i + m_i)^2. \tag{11}$$

The first inequality is obvious. The second follows from (1). The third follows from the fact that the number of atom positions available at processor i is at most $n_i + m_i$, so that the number of forces that can be computed at this processor is bounded by

$$\binom{n_i + m_i}{2} < (n_i + m_i)^2.$$

We say that a set $L \subset A$ *c-isolates* set $M \subset A$ if, whenever $(a, b) \in F,\ a \in M,\ b \notin M$, then either $a \in L$ or $b \in L$. We define the *c-surface* of set M, $\mathrm{S_c}(M)$, to be the least size of a set that c-isolates M.

Theorem 1. $m_i \geq \mathrm{S_c}(A(i))$.

Proof. Let L consist of all atoms a such that either processor i receives a force that acts on a or processor i receives the position of a. Clearly, $m_i \geq |L|$. Let a and b be two atoms such that $(a, b) \in F$, $a \in A(i)$, $b \notin A(i)$. If processor i computes the force between a and b, then it must receive the position of b. Otherwise, processor i must receive from another processor a sum of forces acting on atom a that contains the force $\mathbf{f}_{ab}$. Thus either $a \in L$ or $b \in L$. It follows that L c-isolates $A(i)$, so that $|L| \geq \mathrm{S_c}(A(i))$. □

We can thus prove lower bounds on communication by proving lower bounds on the surface of the set of atoms maintained by each processor.

Theorem 2. *Let δ and ε be the constants defined in the constant density assumption. There exist positive constants k, η, and λ (that depend only on δ and ε) so that the following holds. Let M be a set of $m = |M| \leq \eta n$ atoms. If $\delta \leq c/2k$, then*

$$\mathrm{S_c}(M) \geq \lambda \min(m, m^{2/3} n^{1/3} c/d).$$

Before we prove this theorem, we show how it can be used to derive lower bounds on communication. We restrict ourselves to algorithms that are load balanced:

$$\forall i, \qquad f_i \leq 2(\tfrac{4}{3})\pi(c/d)^3 n^2/p.$$

(The constant 2 is arbitrary—any constant > 1 will do.)

Theorem 3 (Low Parallelism). *There exists a constant p_0 such that if $p_0 < p < (d/c)^3$, then*

$$\max_i m_i > \lambda(c/d)n/p^{2/3}.$$

Proof. Since $\sum_i n_i \geq n$, then $n_i \geq n/p$, for some i. There are two cases to consider.

Case A: $\exists\, i$ *such that* $n/p \leq n_i \leq \eta n$. Then, by Theorem 2,

$$m_i \geq \lambda \min(n/p,\ (n/p)^{2/3} n^{1/3}(c/d)) \ =\ \lambda(c/d)n/p^{2/3}.$$

Case B: $\exists\, i$ *such that* $n_i > \eta n$. As the algorithm is load balanced, then processor i computes at most $2(\frac{4}{3})\pi(c/d)^3 n^2/p$ forces. Since each atom interacts with at least $\frac{4}{3}\pi(1-\varepsilon)(c/d)^3 n$ other atoms, processor i does not compute all forces acting on at least

$$\eta n - \frac{2(\frac{4}{3})\pi(c/d)^3 n^2/p}{\frac{4}{3}\pi(1-\varepsilon)(c/d)^3 n} = n\left(\eta - \frac{2}{(1-\varepsilon)p}\right)$$

atoms in $A(i)$. Thus

$$m_i \geq n\left(\eta - \frac{2}{(1-\varepsilon)p}\right) \geq \frac{\lambda n}{p} \geq \frac{\lambda(c/d)n}{p^{2/3}},$$

for large enough p. □

The lower bound in the theorem above matches the upper bound of (8), within a constant factor. Thus, the hybrid algorithm generates a minimal communication volume, in the low parallelism case.

Theorem 4 (High Parallelism). *There exist a positive constant α such that if $p \geq (d/c)^3$, then*

$$\max_i m_i > \alpha(c/d)^{3/2} n/p^{1/2}.$$

Proof.

Case A: $\exists\, i$ *such that* $n_i \geq (c/d)^{3/2} n/p^{1/2}$. Then

$$\begin{aligned} m_i &\geq \lambda \min(n_i, n_i^{2/3} n^{1/3}(c/d)) \\ &\geq \lambda \min((c/d)^{3/2} n/p^{1/2}, (c/d)^2 n/p^{1/3}). \end{aligned}$$

However, $(c/d)^{1/2} p^{1/6} \geq 1$, so that $(c/d)^2 n/p^{1/3} \geq (c/d)^{3/2}/p^{1/2}$ and

$$m_i \geq \lambda(c/d)^{3/2} n/p^{1/2}.$$

Case B: $\forall i, n_i < (c/d)^{3/2} n/p^{1/2}$. Then (10) and (11) imply that

$$\max_i (n_i + m_i)^2 \geq \max_i f_i \geq \tfrac{4}{3}\pi(1-\varepsilon)(c/d)^3 n^2/p$$

so that

$$\begin{aligned}\max_i m_i &\geq (\tfrac{4}{3}\pi(1-\varepsilon))^{1/2}(c/d)^{3/2}n/p^{1/2} - (c/d)^{3/2}n/p^{1/2}\\ &= ((\tfrac{4}{3}\pi(1-\varepsilon))^{1/2} - 1)(c/d)^{3/2}n/p^{1/2}.\end{aligned}$$

The claim follows, with $\alpha = \min(\lambda, ((\frac{4}{3}\pi(1-\varepsilon))^{1/2} - 1))$. □

The lower bound given in the last theorem matches the upper bound given in (7), within a constant factor. Thus, the hybrid algorithm generates a minimal communication volume, in the high parallelism case.

We now turn to the proof of Theorem 2. We first prove a continuous version of this theorem, for Euclidian spaces, and next derive from it the discrete theorem.

We first introduce several definitions. We denote by $\mathrm{Vol}(U)$ the *volume* (i.e., Lebesgue measure) of a set $U \subset \mathbb{R}^n$. This is well defined when U is compact.

We denote by B_r^n the ball of radius r centered at the origin in $\mathbb{R}^n$; the superscript n will be suppressed when obvious from the context. We have $\mathrm{Vol}(B_r^n) = v_n r^n$, where

$$v_n = \mathrm{Vol}(B_1^n) = \begin{cases}\pi/m! & \text{for} \quad n = 2m,\\ 2(2\pi)^m/(2m+1)! & \text{for} \quad n = 2m+1.\end{cases}$$

Given two sets A and B in $\mathbb{R}^n$, we denote by $A + B$ the set

$$A + B = \{\mathbf{x} + \mathbf{y} \;:\; \mathbf{x} \in A, \mathbf{y} \in B\}.$$

Note that if A and B are compact, then $A + B$ is compact.

We denote by $B(U, r)$ the set $U + B_r$. It is easy to see that if U is compact then

$$B(U, r) = \{\mathbf{y} \in \mathbb{R}^n \;:\; D(U, \mathbf{y}) \leq r\}.$$

We use the following inequality:

Theorem 5 (Brunn–Minkowski Inequality). *Let A and B be two compact sets in $\mathbb{R}^n$. Then*

$$\mathrm{Vol}^{1/n}(A + B) \geq \mathrm{Vol}^{1/n}(A) + \mathrm{Vol}^{1/n}(B).$$

Equality obtains only in the following three cases: (1) $\mathrm{Vol}(A + B) = 0$; (2) *A or B consists of a single point*; (3) *A and B are similar.*

The reader is referred to Chapter 2 of [5] for a proof to this theorem.

Corollary 1. *Let U be a nonempty compact set in $\mathbb{R}^n$. Let r be the radius of a ball with the same volume as U, i.e.*, $\mathrm{Vol}(U) = \mathrm{Vol}(B_r)$. *Then, for any $c > 0$*, $\mathrm{Vol}(B(U, c)) \geq \mathrm{Vol}(B_{r+c})$; *equality obtains only if U is a ball.*

This corollary is analogous to the well-known isoperimetric theorem that states that a ball has the minimum surface for a given volume. Here, we consider the volume of a shell of a fixed thickness around a body of a given volume, and claim that the volume of the shell is minimized when it encases a ball. The classical isoperimetric result can be derived from this corollary by considering the limit when the thickness of the shell goes to zero; see [5].

Proof. We have, by the Brunn–Minkowski inequality,

$$\begin{aligned}\mathrm{Vol}^{1/n}(B(U,c)) &= \mathrm{Vol}^{1/n}(U+B_c) \geq \mathrm{Vol}^{1/n}(U) + \mathrm{Vol}^{1/n}(B_c)\\ &= \mathrm{Vol}^{1/n}(B_r) + \mathrm{Vol}^{1/n}(B_c) = v_n^{1/n} r + v_n^{1/n} c = v_n^{1/n}(r+c)\\ &= \mathrm{Vol}^{1/n}(B_{r+c}).\end{aligned}$$

Equality obtains if and only if U is similar to B_c, i.e., if and only if U is a ball. □

Let U, V be subsets of $\mathbb{R}^n$. We say that V *c-isolates* U if

$$\forall\, \mathbf{x} \in U, \quad \forall\, \mathbf{y} \notin U, \qquad D(\mathbf{x},\mathbf{y}) < c \rightarrow \mathbf{x} \in V \vee \mathbf{y} \in V.$$

We define the *c-surface* $\mathrm{S_c}(U)$ of a compact set $U \subset \mathbb{R}^n$ to be the least volume of a set that c-isolates U.

Theorem 6. *Let U be a compact set of volume $\mathrm{Vol}(U) = v_n r^n$. Then*

$$\mathrm{S_c}(U) \geq v_n(r^n - \max(0, (r-c)^n)).$$

That is, the volume of a set V that c-isolates U is minimized when U is a ball and V is a shell extending from the surface of the ball to a depth of c.

Proof. Let V be a set that c-isolates U. Since V c-isolates $V \cup U$ we can assume without loss of generality that $V \subseteq U$. Let $U_1 = \{\mathbf{x} \in U \;:\; \forall \mathbf{y} \notin U,\, D(\mathbf{x},\mathbf{y}) \geq c\}$ be the set of points in U that are at distance at least c from points outside U. If $U_1 = \emptyset$, then each point in U is at distance $< c$ from U^{c}, so that $V = U$, and $\mathrm{Vol}(V) = \mathrm{Vol}(U) = v_n r^n$.

Otherwise, if $U_1 \neq \emptyset$, let $U_2 = B(U_1, c)$. Let $\mathrm{Vol}(U_1) = v_n r_1^n$ and let $\mathrm{Vol}(U_2) = v_n r_2^n$. By the previous corollary, $r_2 \geq r_1 + c$. It is easy to see that $V \supseteq U - U_1$ and $U \supset U_2$, so that $r \geq r_2 \geq r_1 + c$. Thus

$$\mathrm{Vol}(V) \geq \mathrm{Vol}(U) - \mathrm{Vol}(U_1) = v_n(r^n - r_1^n) \geq v_n(r^n - (r-c)^n). \qquad \square$$

We now return to the proof of Theorem 2

Proof. We need to prove that if L c-isolates M, $|L| = \ell$, and $|M| = m \leq \eta n$, then

$$\ell \geq \lambda \min(m, m^{2/3} n^{1/3}(c/d)). \tag{12}$$

We prove the theorem for the special case where $L \subseteq M$; this entails the general case, with new constants $\eta' = \frac{2}{3}\eta$ and $\lambda' = \min(\frac{1}{2}, \lambda)$. Indeed, if $\ell > m/2$, then the claim follows directly. If, on the other hand, $\ell < m/2$, then L isolates $L \cup M$ and $|L \cup M| < \frac{3}{2}m \leq \frac{3}{2}\eta' n = \eta n$, so that the conditions of the special theorem apply.

We say that a set $U \subset \mathbb{R}^3$ is *α-rectifiable* if there is a disjoint set of cubes $\{\hat{C}_1, \ldots, \hat{C}_r\}$ such that $U \subseteq \bigcup \hat{C}_i$ and a disjoint set of cubes $\{\breve{C}_1, \ldots, \breve{C}_s\}$ such that $U \supseteq \bigcup \breve{C}_i$, where all cubes are of dimension $\geq \alpha$, and

$$2\sum_{i=1}^{3} \mathrm{Vol}(\breve{C}_i) \geq \mathrm{Vol}(U) \geq \frac{1}{2}\sum_{i=1}^{r} \mathrm{Vol}(\hat{C}_i).$$

(The constant 2 in the definition is arbitrary and can be replaced by any constant > 1.)

If U is a δ-rectifiable set, then the number of atoms with positions in U is at least $0.5(1-\varepsilon)n\mathrm{Vol}(U)/d^3$ and at most $2(1+\varepsilon)n\mathrm{Vol}(U)/d^3$.

We first prove the corollary ignoring periodicity: we assume an unbounded system of distinct atoms that fulfill the constant density assumption, and define $F = \{(a, b) : D(a, b) < c\}$. Thus, if $a \in M$, $b \notin M$, and $D(a, b) < c$, then $a \in L$.

Let

$$U = \bigcup_{a \notin M} B(\mathbf{x}(a), k\delta),$$

where $k > 1$ is a constant to be defined later. Let

$$V = B(U, c - k\delta) - U.$$

One can verify that the sets U^{c} (the complement of U) and V are bounded; they are δ-rectifiable, for k large enough (independent of the atom numbers and locations) and for $c \geq 2k\delta$. This defines k.

Let b be an atom such that $\mathbf{x}(b) \in V$. By the definition of V, we have $\mathbf{x}(b) \notin U$. This implies, by the definition of U, that $b \in M$.

By the definition of V, there exists a point $\mathbf{y} \in U$ such that $D(\mathbf{x}(b), \mathbf{y}) < c - k\delta$. However, if $\mathbf{y} \in U$, then there exists an atom $a \notin M$ such that $D(\mathbf{x}(a), \mathbf{y}) < k\delta$. Thus $D(\mathbf{x}(a), \mathbf{x}(b)) < c$. It follows that $b \in L$. Thus all atoms with positions in V are in L. It follows that

$$\ell \geq \tfrac{1}{2}(1-\varepsilon)n\mathrm{Vol}(V)/d^3. \tag{13}$$

Let $M_1 = M - L$ and let $m_1 = m - \ell = |M_1|$. If $a \in M_1$, then $a \notin L$, so that $\mathbf{x}(a) \notin V$. Since L c-isolates M_1, then for any atom $b \notin M$, $D(a, b) > c > k\delta$. This implies that $\mathbf{x}(a) \notin U$. Thus, U^{c} contains the positions of all atoms in M_1. It follows that

$$m - \ell \leq 2(1+\varepsilon)n\mathrm{Vol}(U^{\mathrm{c}})/d^3. \tag{14}$$

By definition, V $(c-k\delta)$-isolates U, hence $(c-k\delta)$-isolates U^{c}. It follows, by Theorem 6, that

$$\mathrm{Vol}(V) \geq \tfrac{4}{3}\pi(r^3 - \max(0, (r - c + k\delta)^3)), \tag{15}$$

where $\mathrm{Vol}(U^{\mathrm{c}}) = \frac{4}{3}\pi r^3$. The result now follows by putting (13), (14), and (15) together. We have

$$\begin{aligned}\ell &\geq \tfrac{1}{2}(1-\varepsilon)n\tfrac{4}{3}\pi(r^3 - \max(0, (r-c+k\delta)^3))/d^3 \\ &= \tfrac{2}{3}(1-\varepsilon)\pi n(r^3 - \max(0, (r-c+k\delta)^3))/d^3\end{aligned}$$

and

$$m - \ell \leq 2(1+\varepsilon)n\tfrac{4}{3}\pi r^3/d^3 = \tfrac{8}{3}(1+\varepsilon)\pi n r^3/d^3.$$

We consider three cases:

Case A: $\ell \geq m/2$. Then (12) follows, if $\lambda \leq \frac{1}{2}$.

Case B: $\ell < m/2$ *and* $r \leq 2(c-k\delta)$. Then $r - (c-k\delta) \leq r/2$, so that

$$\ell \geq \tfrac{2}{3}(1-\varepsilon)\pi n\tfrac{7}{8}r^3/d^3 = \tfrac{7}{12}(1-\varepsilon)\pi n r^3/d^3.$$

Also,

$$m/2 < m - \ell \leq \tfrac{8}{3}(1+\varepsilon)\pi n r^3/d^3,$$

so that

$$\frac{nr^3}{d^3} \geq \frac{m}{\frac{16}{3}(1+\varepsilon)\pi}$$

and

$$\ell \geq \tfrac{7}{12}(1-\varepsilon)\pi\frac{m}{\frac{16}{3}(1+\varepsilon)\pi} = \frac{7(1-\varepsilon)}{64(1+\varepsilon)}m.$$

Equation (12) follows if

$$\lambda \leq \frac{7(1-\varepsilon)}{64(1+\varepsilon)}.$$

Case C: $\ell < m/2$ *and* $r > 2(c-k\delta)$. Let $f(x) = 3 - 3x + x^2$. Then $f'(x) = -3 + 2x < 0$, if $x < 1.5$. Thus, $f(x)$ is monotonically decreasing in the interval $0 < x < 1.5$. Thus, since $0 < (c-k\delta)/r < 0.5$, then

$$\begin{aligned}r^3 - (r-(c-k\delta))^3 &= r^2(c-k\delta)\left(3 - 3\frac{c-k\delta}{r} + \left(\frac{c-k\delta}{r}\right)^2\right) \\ &> r^2(c-k\delta)(3 - 3(0.5) + (0.5)^2) = \tfrac{7}{4}r^2(c-k\delta) > \tfrac{7}{8}r^2 c.\end{aligned}$$

Thus,

$$\ell > \tfrac{2}{3}(1-\varepsilon)\pi n \tfrac{7}{8} r^2 c/d^3 = \tfrac{7}{12}(1-\varepsilon)\pi n r^2 c/d^3$$

and

$$m/2 < m - \ell \leq \tfrac{8}{3}(1+\varepsilon)\pi n r^3/d^3$$

so that

$$r^3 > \frac{md^3}{\frac{16}{3}(1+\varepsilon)\pi n}$$

and

$$\ell > \tfrac{7}{12}(1-\varepsilon)\pi n \left(\frac{md^3}{\frac{16}{3}(1+\varepsilon)\pi n} \right)^{2/3} c/d^3 = \frac{7(1-\varepsilon)\pi^{1/3}}{48(\frac{2}{3}(1+\varepsilon))^{2/3}} (c/d) n^{1/3} m^{2/3}.$$

Equation (12) follows if

$$\lambda \leq \frac{7(1-\varepsilon)\pi^{1/3}}{48(\frac{2}{3}(1+\varepsilon))^{2/3}}.$$

Thus, the theorem holds, in the aperiodic case, with

$$\lambda = \min\left(\frac{1}{2}, \frac{7(1-\varepsilon)}{64(1+\varepsilon)}, \frac{7(1-\varepsilon)\pi^{1/3}}{48(\frac{2}{3}(1+\varepsilon))^{2/3}} \right).$$

We turn now to the cyclic case. We have $L \subset M$ that c-separates M using the original definition: if $a \in M$, $b \notin M$, $\tilde{D}(a,b) < c$, then $a \in L$.

Let $B \subseteq M$ be the set of atoms in M that are at distance $< c$ from the boundaries of the cell $\mathbf{D}$. We assume, for the time being, that

$$b = |B| \leq 6m(c/d). \tag{16}$$

The set $B \cup L$ c-isolates M in the aperiodic space, where all atom copies are considered to be distinct. It follows that

$$\ell + b \geq \lambda \min(m, m^{2/3} n^{1/3}(c/d)). \tag{17}$$

If

$$\ell \geq (\lambda/2)\min(m, m^{2/3} n^{1/3}(c/d)),$$

then we are done, with a constant $\lambda' = \lambda/2$. Otherwise, we have

$$6m(c/d) \geq b \geq (\lambda/2)\min(m, m^{2/3} n^{1/3}(c/d)).$$

We consider two cases.

Case A: $6(c/d) < \lambda/2$. Then

$$6m(c/d) \geq (\lambda/2)m^{2/3}n^{1/3}(c/d)$$

which implies

$$m \geq (\lambda/12)^3 n.$$

This case is ruled out if we pick $\eta < (\lambda/12)^3$.

Case B: $6(c/d) > \lambda/2$. Pick an atom $a \in M - L$ (if there are none, then $\ell = m$, and we are done). The ball $B_c(\mathbf{x}(a))$ contains at least $\frac{4}{3}\pi(1-\varepsilon)(c/d)^3 n$ atoms; all of these atoms are in M. Thus

$$m \geq \tfrac{4}{3}\pi(1-\varepsilon)(c/d)^3 n > \tfrac{4}{3}\pi(1-\varepsilon)(\lambda/12)^3 n.$$

This case is ruled out if we pick

$$\eta < \tfrac{4}{3}\pi(1-\varepsilon)(\lambda/12)^3.$$

We assumed that $|B| \leq 6m(c/d)$. We now reduce the general case to a case where this assumption holds. Since $|M| \leq m$, there must be a horizontal slice of the cell **D**, of thickness $2c$, that contains no more than $m(2c/d)$ atoms from M. We can cyclically shift all atoms in cell **D** so that this slice is centered around the horizontal face of **D**, with no change in the relative positions of the atoms. The same applies to each of the other two dimensions. After the shifts, we have $|B| \leq 6m(c/d)$. □

6. Conclusion

We have defined in this paper a new algorithm for a direct solution of the N-body problem with cutoff, and have shown that this algorithm is optimal, if work is kept balanced.

The algorithm was defined for a parallel computer with a complete communication graph, where each processor can communicate directly with each other. However, the algorithm uses only communication along the axes of a three-dimensional mesh and most communications are among nearest neighbors in a three-dimensional cyclic mesh. Therefore, the hybrid algorithm can be implemented efficiently on mesh connected computers. A detailed analysis of the algorithm for a computer with a three-dimensional cyclic mesh topology is provided in [12].

The lower bound proofs assumed that the location(s) where each force is computed and each atom coordinates are updated are fixed. We suspect that the bounds hold even if these locations are allowed to change, but have no formal proof.

The lower bound on communication is not valid if work is not balanced—in particular, one can perform all the computation on one processor, with no communication.

It is likely that a tradeoff can be shown between computation and communication. It would be nice to generalize the lower bound results so as to optimize this tradeoff, for any particular value of ℓ and τ.

The lower bounds were shown for a restricted computation model, that prohibits methods such as the multipole algorithm. Indeed, the multipole algorithm has better (asymptotic) communication complexity than implied by our lower bound. However, it is likely that that the lower bound holds under weaker constraints. We conjecture that it holds even if one allows arbitrary linear combinations of coordinates or forces in the computation. The lower bounds on communication would then follow from dimension arguments.

Acknowledgments

We thank Jose Castanos and Don Coppersmith for carefully reading an early version of this paper, and Miklos Ajtai, Isaac Chavel, and especially Michael Schub for offering help with the geometric inequalities of Section 5.

References

[1] M. P. Allen and D. J. Tildesley. *Computer Simulation of Liquids*. Oxford Science Publications, Oxford, 1987.

[2] G. S. Almasi, C. Caşcaval, J. G. Castaños, M. Denneau, W. Donath, M. Eleftheriou, M. Giampapa, H. Ho, D. Lieber, J. E. Moreira, D. Newns, M. Snir, and Jr. H. S. Warren. Demonstrating the scalability of a molecular dynamics application on a petaflop computer. *International Journal of Parallel Programming*, 30(4):317–351, 2002.

[3] A. Bar-Noy and S. Kipnis. Designing broadcasting algorithms in the postal model for message-passing systems. In *Proceedings of the* 4*th Annual Symposium on Parallel Algorithms and Architectures*, pages 13–22, 1992.

[4] J. Board and K. Schulten. The fast multipole algorithm. *IEEE Computational Science & Engineering*, 2:56–59, 2000.

[5] Yu. D. Burago and V. A. Zalgaller. *Geometric Inequalities*. Springer-Verlag, Berlin, 1988.

[6] M. T. Dickerson and D. Eppstein. Algorithms for proximity problems in higher dimensions. *Computational Geometry Theory and Applications*, 5:277–291, 1996.

[7] P. Ewald. Die Berechnung optischer und elektrostatischer Gitterpotentiale. *Annalen der Physik*, 64:253–287, 1921.

[8] L. F. Greengard. *The Rapid Evaluation of Potential Fields in Particle Systems*. MIT Press, Cambridge, MA, 1988.

[9] L. Greengard and V. Rokhlin. A fast algorithm for particle simulation. *Journal of Computational Physics*, 73(2):325–348, 1987.

[10] R. Motwani and P. Raghavan. *Randomized Algorithms*. Cambridge University Press, Cambridge, 1995.

[11] S. Plimpton. Fast parallel algorithms for short-range molecular dynamics. *Journal of Computational Physics*, 117:1–19, 1995.

[12] M. Snir. A note on n-body computation with cutoffs. Technical Report RC22059, IBM T. J. Watson Research Center, 2001.

[13] V. E. Taylor, R. Stevens, and K. Arnold. Parallel molecular dynamics: implications for massively parallel machines. *Journal on Parallel and Distributed Computing*, 45(2):166–175, September 1997.

Received February 18, 2002, *and in revised form July* 14, 2003, *and in final form July* 31, 2003.
Online publication January 23, 2004.