Nearest Neighbor Trees and N-body Simulation

Richard Anderson
Department of Computer Science and Engineering
University of Washington*

Abstract

In this paper, we study data structures for N-body simulation. The basic data structure is a hierarchical partition of the input which is used for clustering particles in order to perform an efficient force computation. We are interested in Nearest Neighbor Trees which are formed by repeatedly combining mutually closest pairs of points to create a binary tree. This data structure is an alternative to the more commonly used oct-tree structure, and may be advantageous in practice. In this paper, our goal is to establish a theoretical basis to support the use of Nearest Neighbor Trees. Our main result is to prove that Strict Nearest Neighbor Trees satisfy a logarithmic height bound. We also establish various performance bounds on the N-body computation using Strict Nearest Neighbor Trees.

*Sabbatical Address: Department of Computer Science and Automation, Indian Institute of Science, Bangalore, 560012, India. Email: anderson@csa.iisc.ernet.in

1 Introduction

In this paper, we study a computational geometry problem that is motivated by N-body simulation. The current algorithm of choice for astrophysical simulation is the Particle-Cluster algorithm. Our goal is to prove that a Particle-Cluster algorithm based upon Nearest Neighbor Trees is as good as the more commonly used Barnes-Hut algorithm[BH86]. The Nearest Neighbor algorithm was developed independently by Benz et al. [BBCP90] and by Jernigan and Porter [JP89], and has been observed to run very well in practice [Mak90]. Heuristic arguments have been given to explain the good performance, but up until now there has been no rigorous proofs that it is a fast algorithm.

We begin with a discussion of N-body simulation in Astrophysics and present the generic Particle-Cluster algorithm as well as the Barnes-Hut algorithm. We then introduce Nearest Neighbor Trees and Strict Nearest Neighbor Trees. The main result of the paper is to establish a logarithmic bound on the height of Strict Nearest Neighbor Trees. This allows us to show that the Strict Nearest Neighbor tree based algorithm has a run time that is within a logarithmic factor of the Barnes-Hut algorithm. We give evidence that this logarithmic factor can be removed. We conclude with a short discussion of general Nearest Neighbor trees and on possible extensions of this work.

Astrophysical Simulation N-body simulation is a fundamental tool used in astrophysical research. The basic approach is to follow the time evolution of a set of particles under gravitational force. This is done by computing the force upon each particle, and then updating the positions and velocities with a simple integration. A sufficiently large number of time steps is used to be able to observe the dynamics of the system. The run time of the simulation algorithm is dominated by the cost of computing the force on each particle. The problem of computing the forces is generally referred to as the N-body problem.

In astrophysics it is critical to be able to study systems with a very large number of particles. The largest simulations to date have followed about 20 million particles for one thousand time steps [SW92a]. Astrophysicists argue that qualitatively different problems could be addressed by simulation if the number of particles were increased by one or two orders of magnitude. There is currently substantial research work going on to enable these massive simulations.

N-body algorithms have been improved dramatically in the last two decades. The force computation on a single particle is the summation of n terms, so there is a direct algorithm for the problem which runs in $O(n^2)$ time. Improvements on this run time have come by computing an approximation of the forces instead of exact forces. The particle-cluster algorithm computes the force on each particle by using a spatial hierarchy to cluster particles. The algorithm was independently developed by several researchers [App85, BH86, JP89], with the algorithm due to Barnes and Hut being the one that is currently most widely used. The run time for the particle-cluster algorithm is generally considered to be $O(n \log n)$. The Greengard-Rokhlin algorithm [GR87, Gre87] is a cluster-cluster algorithm which achieves $O(n)$ run time by computing forces between clusters. Although the Greengard-Rokhlin algorithm is

elegant and asymptotically superior, it is not widely used in astrophysical simulation because of the algorithm's complexity and large constant factors.

Particle-Cluster Algorithm The particle-cluster algorithm is to compute the force on each particle by considering a partition of the particle set into clusters. The force is computed by approximating the contribution of each cluster. A recursive formulation of the algorithm is given below. The routine computes the force on p by traversing a tree which represents a hierarchy of the particles. (For ease of exposition, the tree is binary, although it could have a higher degree.) The subroutine $GoodApprox(p, T)$ tests whether or not the approximation of the force of T on p passes the accuracy threshold. *ApproxForce* and *ExactForce* are used to compute the forces.

```
Evaluate (p : ParticleType, T : TreeNode)
    if IsLeaf(T)
        return ExactForce(p, T);
    if GoodApprox(p, T)
        return ApproxForce(p, T);
    return Evaluate(p, T.left) + Evaluate(p, T.right)
```

In principle, any tree which gives a hierarchical partition of the particle set could be used in the algorithm. However, to achieve reasonable performance, the clusters should be localized, and the trees should have shallow depth. The Barnes-Hut algorithm is based on using an Oct-Tree. In this paper, we consider a competing choice, the Nearest Neighbor tree. A Nearest Neighbor tree is built in a bottom up manner, with the hope that its structure will more accurately match the geometry of the point set than an Oct-Tree does.

Practical Considerations The particle-cluster algorithm is widely used in astrophysical simulation. It is important to consider how the algorithm is used in practice so that the algorithm questions that we address are relevant. The following points are important to our study:

- The prime interest is in increasing the number of particles used. This runs up against both CPU and memory limitations.
- The computation is done to a low precision. The accuracy controls are generally set so that the algorithm is operating outside of its "safe regime".
- The particle distribution is often highly non-uniform and exhibits a high degree of clustering.
- The total number of force computations dominates the cost of the algorithm. The cost can be measured by counting the number of calls to *ApproxForce* and *ExactForce*, (with the *ApproxForce* computations being more expensive).
- The cost of tree construction if very small, figures cited for the Barnes-Hut algorithm range from 2% to 4% of the total run time [Sal90, Her87].
- Achieving a constant factor improvement in run time would be of substantial practical interest.

Data Structures One way to improve the performance of the particle-cluster algorithm is to find a better data structure. The data structure determines the number of force computations that are performed. Since the force computations are the dominant cost of the algorithm, we can measure the performance of the data structure by counting the number of force comparisons. The hypothesis is that a data structure which matches the geometry of the point set will allow the computation to be done with a smaller number of force computations. Since the cost of tree construction is minor, it is possible to use a data structure which is more expensive to construct than the oct-tree and still achieve a net gain in performance. (There are also opportunities to use incremental data structures, where the construction cost can be amortized across time steps of the simulations. These techniques are not necessary for oct-trees, but could be used to reduce the construction cost of a more complicated data structure.)

There are several properties that the spatial data structure should have in order to lead to efficient computations. First of all, it should have shallow depth. In particular, it should have logarithmic depth assuming reasonable particle distributions. Second, it should have a locality principle, so that the particles associated with the node of the tree are within a contiguous region of space. Finally, the tree structure should match the geometry of the point set. This means that clusters of particles should correspond to nodes of the tree. The oct-tree data structure is good for the first two points, but does not necessarily match the structure of the points set. Since the oct-tree is constructed by choosing splitting planes oblivious to particle distribution, it is common for a tightly packed cluster to be divided between separate tree nodes.

The candidate data structure to use instead of oct-trees is the nearest neighbor tree. This data structure was independently introduced by Benz et al. [BBCP90], and by Jernigan and Porter [JP89]. Experimental studies [Mak90] have shown that this data structure is competitive with oct-trees, although results are not conclusive. The goal of this paper is to develop a theoretical justification for using nearest neighbor trees. Up until now, only heuristic arguments have been given to show that nearest neighbor trees behave well. We are able to establish that one version of nearest neighbor trees satisfies a logarithmic height bound (with assumptions on particle distribution and mass), and we can also relate the tree height to performance of the N-body algorithm. However, there are still important open problems to solve to generalize the height bounds to the full class of Nearest Neighbor Trees, and also to gain a deeper understanding of the run time and accuracy of the N-body algorithm with Nearest Neighbor Trees.

Unless noted otherwise, our results are for three dimensions, which is the prime case of interest in simulation. Many of the results generalize to any constant dimension, although we do not attempt to state them in the more general form.

2 Barnes-Hut Algorithm

Since we will want to compare the Nearest Neighbor Tree based algorithm with the Barnes-Hut algorithm, it is necessary to give a little more detail on Barnes-Hut. As mentioned above, the Barnes-Hut algorithm is the generic algorithm with an oct-tree data structure. The test $\mathit{GoodApprox}(p, T)$, compares the distance from p to T with the

size of the region enclosing the particles of T. If d is the distance from p to the center of mass of T, and s is the side length of the bounding box of T, the approximation is accepted if $\frac{s}{d} < \theta$ where θ is the user supplied accuracy parameter. The values of θ used in practice are in the range $(0.7, 1.0)$. These values are are high enough that the worst case guarantees are very weak, although observed behavior is far better.

Although the run time for the Barnes-Hut algorithm is generally cited as $O(n \log n)$, the real run time depends upon the distribution of the particles and the height of the tree. In an oct-tree, the tree can have an arbitrarily large height for a fixed number of particles, so the run time is actually unbounded. If a tree such as a *fair-split* tree is used [CK92], then the height is bounded by $O(n)$. (However, these pathological examples are of no interest to astrophysicists, since the data is not represented with sufficient precision for these bad cases to occur.) It is possible to give a run time bound for the Barnes-Hut algorithm in terms of the sum of the leaf depths of the tree, which does give the $O(n \log n)$ time bound for trees which are reasonably balanced. The following two lemmas, given without proof, lead fairly directly to the performance bound for the algorithm.

Lemma 1 *Let x be a particle. The number of cells of size D encountered when evaluating the force on x is bounded by a constant.*

Lemma 2 *Let C be a cell of size D. The number of particles x located in leaf cells of size $D' > D$ which encounter C is bounded by a constant.*

Theorem 1 *Let T be an oct-tree with total leaf depth L. The Barnes-Hut algorithm takes time $O(L)$ on input T to compute the force on all of the particles.*

3 Nearest Neighbor Trees

Nearest neighbor trees are built bottom up by combining pairs of particles at their center of mass. There are a number of variants of nearest neighbor trees, depending upon the details of the algorithm used for construction. The key is to only combine mutually nearest neighbors.

Definition 1 *Points p and q are mutually nearest neighbors if p is a nearest neighbor of q and q is a nearest neighbor of p.*

The most general method for defining a nearest neighbor tree is to allow an arbitrary subset of the nearest neighbor pairs to be merged. It is most convenient to define nearest neighbor trees in terms of a construction algorithm.

We use the following notation. $Mass(p)$ denotes the mass of particle p, and $Dist(p, q)$ the distance between particles p and q. We use $p \odot q$ for the result of merging p and q at their center of mass.

NearestNeighbor(S)
  while $|S| > 1$
    Let $\langle p_1, p_2 \rangle, \ldots \langle p_{2k-1}, p_{2k} \rangle$ be mutually nearest neighbors.
    for $j := 1$ **to** k
      Replace p_{2j-1} and p_{2j} by $p_{2j-1} \odot p_{2j}$.

There is a natural binary tree associated with this process: whenever the node p replaces the nodes p_1 and p_2, we think of p_1 and p_2 as becoming the children of the node p. Since the algorithm is non-deterministic in its choice of nearest neighbor pairs, it can construct many different trees. We define all of them to be nearest neighbor trees.

Definition 2 *A tree is a "nearest neighbor tree" for point set S if it can be constructed by the algorithm NearestNeighbor.*

We can restrict the construction process to merge only a single pair of nodes per step. This reduces the number of trees that can be constructed.

Definition 3 *A tree is a "sequential nearest neighbor tree" for point set S if it can be constructed by the algorithm NearestNeighbor restricted to merging a single pair of particles per iteration.*

The final restriction is to only merge the closest pair of particles per step, getting a deterministic method of construction (assuming a fixed tie breaking rule for equal distances).

Definition 4 *A tree is a "strict nearest neighbor tree" for point set S if it can be constructed by the algorithm NearestNeighbor restricted to merging only the closest pair of particles each iteration.*

It is not difficult to see that these definitions are distinct. The differences between the classes of trees do not appear that large. Both Benz et al., and Jernigan and Porter construct trees by combining a subset of the nearest neighbors, so they use trees of the most general class. We believe that these three classes all have roughly the same properties when used in N-body algorithms. However, for technical reasons, our proofs only apply to strict nearest neighbor trees. The reason for wanting to use the most general class of trees is that construction algorithms might be easier. However, it turns out that strict nearest neighbor trees can be constructed in $O(n \log n)$ time, so even though the results only apply to the smallest class of nearest neighbor trees, it is a case with an efficient construct algorithm.

3.1 Tree Construction

Strict Nearest Neighbor trees can be constructed in $O(n \log n)$ time. This is done by using a data structure that can record nearest neighbor information and supports efficient insertion and deletion of points. We use the Fair Split Tree [CK92] augmented so that it can support incremental operations [CK93]. $O(n \log n)$ is optimal, based upon lower bounds for element distinctness. The question of parallel construction of Nearest Neighbor Trees and Strict Nearest Neighbor Trees is wide open, and is very relevant to this research.

4 Strict Nearest Neighbor Trees

First of all, we would like to show that Strict Nearest Neighbor Trees have limited depth, and then we would like to establish performance bounds for the N-body algorithm based upon the total leaf depth of the tree. We devote most of our effort to establishing a depth bound.

4.1 Lower bounds

Although we would like to establish logarithmic upper bounds, we can give a couple of easy linear lower bounds. In the following one-dimensional examples, $Pos(p)$ give the position of p along the x-axis.

Example 1 *Let* $Pos(p_i) = 2^i$ *and* $Mass(p_i) = 1$.

Example 2 *Let* $Pos(p_i) = \sum_{j=0}^{i}(1 + \frac{2}{n})^j$ *and* $Mass(p_i) = n^i$.

In both of these examples, the resulting tree has depth $n - 1$. Although these examples show that the trees can have large depth, they are not interesting examples to the astrophysicists, since the precision necessary to construct the bad examples is far greater than the precision that is used in practice. However, the conclusion that we must draw from these examples is that our bounds must depend upon the mass and position of the particles.

We define the distance ratio to be

$$\Delta = \frac{\max_{p \neq q} Dist(p, q)}{\min_{p \neq q} Dist(p, q)},$$

and the mass ratio to be

$$M = \frac{\sum_p Mass(p)}{\min_p Mass(p)}.$$

Examples 1 and 2 show that the upper bounds can not be much better than logarithmic in Δ and M.

4.2 Neighborhoods

The main theorem is to bound the depth of strict nearest neighbor trees. The idea of the proof is that as particles get merged together, they get bigger and bigger "neighborhoods". However, it turns out to be a moderate amount of work to convert this idea to a proof because of technical hurdles which must be overcome.

The first difficulty is in defining the neighborhoods of particles. We would like to have a definition such that neighborhoods are strictly increasing in size as particles are merged. However, when particles are merged, points can get closer together. The key case is three equal mass points which are the vertices of an equilateral triangle. If they are initially one unit apart, and a pair of them are merged, the resulting separation is $\sqrt{3}/2$. The concern is that the separation could shrink to an arbitrary small amount. However, we show that this cannot happen. We define the radius of a node to be the closest point distance when it was created.

Definition 5 *If p is a leaf, then $Rad(p)$ is 0, otherwise, if $p = p_1 \odot p_2$, then $Rad(p) = Dist(p1, p2)$.*

Lemma 3 *Suppose q is created after p, then $Rad(q) \geq \sqrt{\frac{2}{3}} Rad(p)$.*

Proof: The idea is that if $p = p_1 \odot p_2$ and for some particle q, $Dist(p, q) < Dist(p_1, p_2)$, then p must immediately be merged with some particle. (Here we depend on using the strictness condition.) Although there can be an arbitrarily long sequence of merges creating particles that have separation less than $Dist(p_1, p_2)$, the nearest neighbor distance cannot drop below $\sqrt{2/3} Dist(p_1, p_2)$. The proof to show that the result holds for $\sqrt{2/3}$ is moderately involved. However it is fairly easy to show that the result holds for $\frac{1}{2}$ which is sufficient for our applications of this lemma. ∎

Since it is convenient for us to have a notion of neighborhood which is increasing, we introduce a new version of the radius.

Definition 6 *For a non-leaf particle p,*

$$Rad^*(p) = \max\{Rad(q) \mid q \text{ was created no later than } p\}.$$

If p is a leaf, then $Rad^*(p) = 0$.

4.3 Main Theorem

We can now give a sketch of our main result. We shall prove the following:

Theorem 2 *Let P be a set of particles with distance ratio Δ and mass ratio M. The height of the strict nearest neighbor tree for P is $O(\log \Delta + \log M)$.*

The idea of the proof is to follow a leaf to root path in the tree. Clearly, mass and radius are increasing along the path, so if p is the parent of q, then $Mass(p) > Mass(q)$ and $Rad^*(p) \geq Rad^*(q)$. We will show that there is a constant K such that if we go up K levels, we will either double the mass or double the radius.

The difficulty in proving this theorem is in showing that if the mass is not increasing substantially, then the radius must grow. (The correct intuition comes from thinking of a particle with infinite mass, and showing that its radius must double after a constant number of merges.) A particle can be the nearest neighbor of only a fixed number of other particles. Thus, after a particle p has merged with these particles, its radius shold have increased. However, it is conceivable that other particles could have been created to replace the particles that were merged with p. The proof relies on a major lemma which bounds the number of merges until the radius increases.

The first lemma that we need is to show that if in a sequence of merges, neither the mass nor the radius doubles, then the particle does not move very far.

Lemma 4 *Let $\hat{p_1}, \ldots, \hat{p_k}$ be ancestors of p with $\hat{p_i} = \hat{p_{i-1}} \odot p_{i-1}$, (we consider $p = \hat{p_0}$), and suppose that $Mass(\hat{p_k}) \leq 2Mass(p)$ and $Rad^*(\hat{p_k}) \leq 2Rad^*(p)$, then $Dist(p, \hat{p_k}) \leq 2Rad^*(p)$.*

Proof: The proof follows directly from the observation that

$$Dist(\hat{p}_i, \hat{p_{i-1}}) \leq \frac{Mass(p_i)}{Mass(p)} 2Rad^*(p).$$

∎

Before we get to the main lemma, we need a little more terminology.

Definition 7 *A point q is terminal with respect to p if it is present when p is created, and a point q is induced with respect to p if it is created after p is created.*

Lemma 5 *Let $\hat{p_1}, \ldots, \hat{p_k}$ be ancestors of p with $\hat{p_i} = \hat{p_{i-1}} \odot p_{i-1}$, (we consider $p = \hat{p_0}$), and suppose that $Mass(\hat{p_k}) \leq 2Mass(p)$ and $Rad^*(\hat{p_k}) \leq 2Rad^*(p)$, then each p_i has a descendent p_i' such that p_i' is terminal with respect to p and $Dist(p, p_i') < \sqrt{12}Rad^*(p)$.*

Proof: For convenience, we assume that p is located at the origin and $Rad^*(p) = 1$. The proof is by contradiction. We assume that p_i has all of its terminal descendents outside of a sphere of radius $\sqrt{12}$. Lemma 4 implies that $Dist(p, p_i) \leq 4$. The idea for this proof is that p_i must have so many descendents that at least one pair of them has separation less than one. This would contradict the creation of p as a merger of the closest pair of points.

To give a little more detail on the mechanics of the proof, we begin by creating spherical bands:

$$B_i = \{z \mid 2\sqrt{i+4} \leq z < 2\sqrt{i+5}\}.$$

These bands are chosen so that the number of descendents of p_i must double between subsequent bands.

P_i must have at least two descendents in B_0. Since the number of descents doubls, and P_i has no terminal descendents inside of radius $\sqrt{12}$, P_i has at least 256 descendents in B_7. It follows that there is a $k \geq 8$ such that B_k has at least $2^7 \left(\frac{4}{3}\right)^{k-7}$ terminal points. At least one pair of these points must have separation less than one. ∎

Corollary 1 *There is a constant K, such that going up K levels in the tree doubles either the mass or the radius.*

Proof: If neither the mass nor radius doubles, then the hypotheses for the previous lemma are satisfied. Each of the p_i has a descendent within distance $\sqrt{12}Rad^*(p)$ of p. Since the minimal separation is $\sqrt{2/3}Rad^*(p)$ the number of points is bounded by a constant, so the number of subtrees is bounded. ∎

The proof gives a bound of $215(\log M + \log \Delta)$. Benz et al. [BBCP90] give a heuristic argument that the average case is $1.8 \log n$ and cite experimental evidence of $1.3 \log n$. Clearly there is ample room to improve our bound, although it does establish that the tree height is logarithmic in the mass and the distance ratio.

5 N-body Algorithms

We have shown that strict nearest neighbor trees have logarithmic height, which provides evidence that they will work well in N-body algorithms. However, we want to show that the run time is related to the total leaf depth. Since the N-body computation is an approximation, it is important to consider accuracy as well a pure run time. N-body algorithms are often used in practice under conditions where the provable error bounds are very weak, which makes comparisons of algorithms difficult.

One of the subroutines of the particle-cluster algorithm is $GoodApprox(x, T)$ which tests whether to use T for the force approximation or to subdivide T. In the Barnes-Hut algorithm, this is done by comparing the ratio of the side length of T with the distance from x to the center of mass of T. Using nearest neighbor trees, there are several options on how to implement this test, in particular, there are several ways to measure the size of the region. The conservative method is to use a bounding sphere, which encloses all of the points of the region. Versions of the bounding sphere test are used by Benz et al. and by Jernigan and Porter. A second choice is to use the radius of the node which is the nearest neighbor distance when the node is created. This could allow some particles of the set to be outside of the radius boundary.

We can bound the size of the bounding sphere in terms of the radius. The proof is related to the proof of Lemma 5.

Lemma 6 *If T is a tree node with $Rad(T) = R$, then all particles in T are within distance $O(R \log M)$ from the center of mass of T.*

This result is somewhat disappointing, but it does allow us to prove a result which looks good if we ignore constant and polylog factors. Using Example 2, we can show that there may be particles as far away as $\Omega(R \frac{\log M}{\log \log M})$ so the lemma is almost tight.

Lemma 7 *If a particle set is represented with a Strict Nearest Neighbor Tree T, with total leaf depth L, the run time of the particle-cluster algorithm using the bounding sphere approximation test is $O(L \log^3 M)$.*

We can clearly improve the speed of the algorithm by using a weaker approximation threshold, such as the radius.

Lemma 8 *If a particle set is represented with a Strict Nearest Neighbor Tree T, with total leaf depth L, the run time of the particle-cluster algorithm using the radius approximation test is $O(L)$.*

Some caution has to be used with this result, since with a weaker approximation, we compute a different result (and it is possible to come up with an arbitrarily fast algorithm which computes an arbitrarily bad result). However, we give some evidence that the approximation given by the radius test is good. First of all, we can show the following.

Lemma 9 *Let T be a nearest neighbor tree. The mass density falls of exponentially with increasing distance from the center of mass.*

This shows that although there may be particles a long ways from the center of mass, these particles represent only a small amount of the mass. Since the gravitational attraction is inverse square, the small amount of mass that is far away should have little impact.

Another way to look at the accuracy is to look at the error on an individual particle. We assume that the radius test is implemented so that if x is inside the cluster T, the test fails. (This is not done in the Barnes-Hut algorithm, which leads to problems [SW92b]). The bad case is when the particle x has a close by particle y which falls into a cluster with center of mass far away from x. This can lead to arbitrarily large errors in the Barnes-Hut algorithm, but we can prove the errors are not as great in the nearest neighbor algorithms.

Lemma 10 *The force computed between particles x and y (when y is shifted to the center of mass of its cluster) is reduced by a factor of at most $O(\log M + \log \Delta)$.*

Lemma 11 *The error the force computation on x when measured in terms of the sum of the forces on x is bounded.*

6 Nearest Neighbor Trees

We would like to show that nearest neighbor trees also have a good depth bound, but have not been successful so far. The main difficulty is that the distances between induced points can decrease, so it has not been possible to argue that separation increases on root to leaf paths in the tree. However, we do believe the result holds. The reason for wanting to use nearest neighbor trees instead of the strict version, is that they may be easier to construct, especially in the parallel case.

Conjecture 1 *Let T be a nearest neighbor tree for particle set P with distance ratio Δ and mass ratio M. The height of T is $O(\log \Delta + \log M)$.*

We have been able to establish the result for special cases such as points points on a line or points on a circle. The proof method in one dimension relies on being able to argue that the separation grows exponentially. We have not been able to extend the result to the two dimensional case.

The prime difficulty in dealing with Nearest Neighbor Trees is that it is difficult to get a notion of the neighborhood increasing in size. A step towards showing that neighborhoods increase in size is to show that it is not possible for an induced point to get too close to another point. We can show the following.

Lemma 12 *Let P be a particle set with distance ratio Δ, and suppose that no particles are within distance one of the particle x. There is an $\epsilon > 0$ such that no induced particle can get within distance $\frac{\epsilon}{\log \Delta}$ without merging with the particle x.*

A major step to proving the conjecture would be to remove the factor of $\frac{1}{\log \Delta}$ in the lemma.

7 Conclusions

In this paper we have studied the performance of N-body algorithms which use a novel geometric data structure. These algorithms have been shown empirically to be competative with the Barnes-Hut algorithm, and there is controversy in the astrophysics community as to which method is superior. We are attempting to gain a theoretical understanding of the issues involved. Up until now, the only bounds on performance have bee heuristic. Our main results show that the N-body algorithm using strict nearest neighbor trees satisfies reasonable asymptotic bounds. Although the data structure is not as general as we would like, it is a structure which can be constructed with an efficient sequential algorithm. The results establish that strict nearest neighbor trees have the desired asymptotic behavior but it is still desirable to improve the results to show that the constants are also relatively small.

There are many open problems still to address. With respect to strict nearest neighbor trees, we would like to have a more rigorous performance analysis and to establish that a radius cutoff is safe to use instead of the more pessimistic bounding sphere cutoff. We would also like to find an efficient parallel algorithm for constructing strict nearest neighbor trees. Of course, one of the major problems driving this research is to show that the more general nearest neighbor tree also has small depth and performs well inside an N-body algorithm. Another area of research is to study more sophisticated N-body algorithms, which rely on methods such as variable time steps and more general methods for determining which nodes to expand. It will be interesting to gain a rigorous understanding of how the data structure methods interact with these other techniques.

References

[App85] A. W. Appel. An efficient program for many-body simulation. *SIAM Journal of Scientific and Statistical Computing*, 6:85–103, 1985.

[BBCP90] W. Benz, R. L. Bowers, A. G. W. Cameron, and W. H. Press. Dynamic mass exchange in doubly degenerate binaries. I. 0.9 and 1.2 $M_\odot$ stars. *The Astrophysical Journal*, 348:647–667, 1990.

[BH86] J. E. Barnes and P. Hut. A hierarchical $O(N \log N)$ force-calculation algorithm. *Nature*, 324:446–449, 1986.

[CK92] P. B. Callahan and S. R. Kosaraju. A decomposition of multi-dimensional point-sets with applications to k-nearest-neighbors and n-body potential fields. In *Proceedings of the 24th ACM Symposium on Theory of Computation*, pages 546–555, 1992.

[CK93] P. B. Callahan and S. R. Kosaraju, 1993. Personal Communication.

[GR87] L. Greengard and V. Rokhlin. A fast algorithm for particle simulations. *Journal of Computational Physics*, 73:325–348, 1987.

[Gre87] L. Greengard. *The Rapid Evaluation of Potential Fields in Particle Systems.* PhD thesis, Yale University, 1987.

[Her87] L. Hernquist. Performance characteristics of tree codes. *The Astrophysical Journal Supplement*, 64:715–734, 1987.

[JP89] J. G. Jernigan and D. H. Porter. A tree code with logarithmic reduction of force terms, hierarchical regularization of all variables and explicit accuracy controls. *The Astrophysical Journal Supplement*, 71:871–893, 1989.

[Mak90] J. Makino. Comparison of two different tree algorithms. *The Journal of Computational Physics*, 88:393–408, 1990.

[Sal90] J. K. Salmon. *Parallel Hierarchical N-body Methods*. PhD thesis, California Institute of Technology, 1990.

[SW92a] J. K. Salmon and M. S. Warren. Astrophysical N-body simulations using hierarchical tree data structures. In *Supercomputing*, pages 570–576, 1992.

[SW92b] J. K. Salmon and M. S. Warren. Skeletons from the treecode closet. Technical report, Los Alamos National Laboratory, 1992.