N-CİSİM ALGORİTMALARININ PERFORMANS KARŞILAŞTIRMASI

Burak İNNER[1] **Fatih Erdoğan SEVİLGEN**[2]
[1]Elektronik-Bilgisayar Eğitimi Bölümü
Teknik Eğitim Fakültesi
Kocaeli Üniversitesi, Kocaeli
[2]Bilgisayar Mühendisliği Bölümü
Mühendislik Fakültesi
Gebze Yüksek Teknoloji Enstitüsü, Gebze, Kocaeli
[1]e-posta:binner@kou.edu.tr [2] e-posta: sevilgen@bilmuh.gyte.edu.tr

Anahtar sözcükler:N-cisim, Barnes-Hut,Hızlı Çok-kutup Algoritması, Quadtree, Octree

ABSTRACT
The N-body problem is to simulate a system of N particles in which each particle has an influence on each other particle. Barnes-Hut and Fast Multipole Method are two algorithms solving N-body problem faster than brute-force methods. Both algorithms use hierarchical tree structures (such as Octree for 3D simulations) for clustering particles and concentrate on influence of a cluster instead of influence of each particle in the cluster. In this study, we implement Barnes-Hut, Fast Multipole Method and brute-force algorithms and compare their performance in terms of running-time and accuracy. We observed that although theoretical running-time of Barnes-Hut algorithm is inferior , it provides better performance even for 1 million particles. We also examined the effect of various parameters such as teta, p and maximum number of particles in a leaf on the performance of each algorithm. Optimum parameter values depend on the number of particles in the system.

1. GİRİŞ
Pek çok bilimsel çalışma fiziksel bir sistemin davranışının incelenmesi konusunu ele alır. Bu fiziksel sistemler genellikle birbirlerini etkileyen cisimlerden oluşmaktadır. Klasik N-cisim problemi, N tane cisimden oluşan bir sistemde, her bir cisme diğer cisimlerin etkilerini ve bu etkilerden dolayı oluşan hareketi hesaplanmaktadır. Bu etki, kütle çekim kuvveti veya elektrostatik kuvveti gibi ters kare kuralını sağlayan kuvvetlerdir. N-cisim problemi ile astrofizik, plazma fiziği, akışkanlar mekaniği ve bilgisayar grafikleri gibi pek çok alanlarda karşılaşılmaktadır. Geleneksel fiziksel sistem uygulamaları dışında sayısal karmaşıklık analizi ve eliptik parçalı diferansiyel denklemlerde benzer yöntemlerle çözülebilmektedir.

Dört ve daha fazla cisim bulunan ve ters kare kuralını sağlayan N-cisim problemleri için bilinen herhangi bir analitik çözüm yoktur. Bu durumda sistem benzetimi popüler bir yaklaşımdır. İstenilen zaman sonunda cisimlerin konumlarının bulunabilmesi için benzetim küçük zaman adımlarında çalıştırılır. Her bir adımda cisimlerin diğerlerine olan etkileri hesaplanır. N tane cisim bulunan bir sistemde toplam N.(N-1) hesap yapılmalıdır. Genel olarak fiziksel sistem simülasyonlarında çok fazla sayıda cismin bulunur. Bu sebeple, N sayısı büyüdüğünde gereken işlem sayısı çok hızlı büyüyecek ve algoritma çok uzun zaman alacaktır. Kuvvetin cisimler arasındaki uzaklığın karesi ile ters orantılı olma özelliğinden yaralanılarak kuvvet daha kısa sürede hesaplayacak algoritmalar geliştirilmiştir. Birbirlerinden uzakta bulunan cisimler gruplandırılarak bu gruplarla hesaplamalar yapılmaktadır ve kuvvet hesabı yapılırken ağaç veri yapısı kullanıldığı için bu yöntemlere hiyerarşik ağaç yöntemleri adı verilmiştir. İlk hiyerarşik olarak cisimleri gruplayarak bu grupları ağaç veri yapısında saklayan ve cisimler arasındaki etkileşimi bu ağaç veri yapısı yardımıyla hesaplayan uygulama Andrew A. Appel [3] tarafından yazılmıştır. Gruplandırma işleminde kullanılacak geometrik şekiller keyfi boyutta düzensiz şekiller olarak seçildiğinden algoritmanın analizi ve gerçeklenmesi oldukça zordur. Ağaç yapısı oluşturulduktan sonra bu yapı üzerinden yapılan işlemler her bir cismin kuvvet hesabı için gerekli işlemleri (dengeli bir ağaç yapısı elde edildiğinde) O(N) 'den ortalama O(logN)'e indirmiştir.

Bu çalışmada hiyerarşik ağaç yöntemlerinden (düzenli dağılım olduğunda) bir adım için kuvvet hesaplamasını O(NlogN) zamanda yapabilen Barnes-Hut[1] ve O(N) zamanda yapabilen Hızlı Çok-Kutup[2] algoritmaları ve bu algoritmaların kullandığı veri yapıları karşılaştırılmıştır.

2. N-CİSİM ALGORİTMALARI
Sistem benzetimi her bir küçük zaman adımı için kuvvetleri hesaplar, kuvvetler hesaplandıktan hızlara ve konumlara geçilir. Cisimler yeni yerlerine konumlandırıldıktan sonra bir sonraki küçük zaman adımı için işlemler tekrarlanır. N-cisim algoritmalarını birbirlerinden ayıran kısım ise kuvvet hesaplamasının

yapıldığı adımdır. Bu adım dışında diğer adımlar aynıdır.

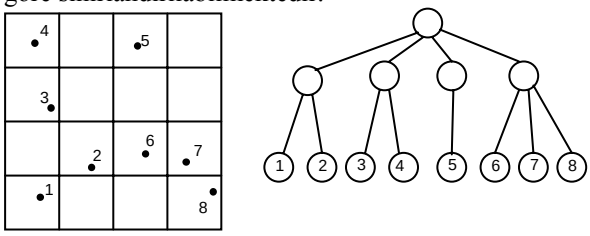
Örnek olarak yerçekimsel kuvvet hesabını ele alalım. Tüm cisimler için yerçekimi kuvveti evrensel çekim kanunu yardımıyla hesaplanmaktadır. Tüm kuvvetler hesaplandıktan sonra gezegenlerin ivmesi hesaplanır. İvmeden hıza, hızdan konumlara geçilir ve gezegenler yeni yerlerine konumlandırılıp işleme devam edilir.

2.1 CİSİM-CİSİM YÖNTEMİ

Bu yöntemde kuvvet hesabı için her bir cisme etki eden kuvvet diğer tüm cisimlerin etkilerinin tek tek hesaplanarak toplanmasıyla (vektörel bileşkesiyle) elde edilir. Bu yöntem en doğru sonucu veren fakat en uzun zaman alan yöntemdir. Bir cisim için gerekli hesaplama sayısı N-1 olduğu için bir adımdaki kuvvet hesabı $O(N^2)$ zamanda gerçekleştirilir. Bu yöntem 10^6 'dan büyük cisim sayıları için günlerce süren bir çalışma zamanından sonra ancak bir adımın hesaplanmasını tamamlamaktadır. Pratikte büyük sayıdaki cisimlerin çözümü için bu yöntem kullanılamaz.

2.2 HİYERARŞİK AĞAÇ YÖNTEMİ

Hiyerarşik ağaç yöntemlerinden Hızlı Çok-kutup ve Barnes-Hut algoritmaları kuvvet hesaplamasını ağaç veri yapısı yardımıyla hesaplamaktadır. Düzenli dağılmış cisimler etkili bir şekilde gruplanarak dengeli bir ağaç yapısı elde edileceğinden işlem zamanı kısacaktır. Düzensiz dağılımlarda ise gruplama işlemi dengesiz bir ağaç yapısına neden olacaktır ve buda algoritmaların işlem zamanlarını ve bellek kullanımlarını artıracaktır. Gruplanmış cisimler ile etkileşim yapılmasından dolayı algoritmalar çok küçükte olsa hatalı sonuçlar üretmiş olacaklardır. Bu hata gruplanmış cisimler ile etkileşim yapılan cisimler arasındaki uzaklığa bağlıdır ve kullanılan algoritmaya göre sınırlandırılabilmektedir.



Şekil 1 Dörtlü Ağaç Yapısı

2.3 AĞAÇ VERİ YAPISI

Barnes-Hut ve Hızlı Çok-Kutup algoritması benzer ağaç yapıları kullanır. Her iki algoritmada her bir zaman adımında ağaç veri yapısını oluşturup işlemlerini bu ağaç veri yapısı yardımıyla gerçekleştirir ve bir sonraki zaman adımında ağaç veri yapısını tekrar oluşturur.

Bu yöntemlerde sistemdeki tüm cisimlerin bulunduğu uzayı içine alan iki boyutta kare, üç boyutta küp olduğu kabul edilir. Bu kare hiyerarşik olarak iki boyutlu sistem için dört, üç boyutlu sistem için sekiz eşit parçaya bölünür. İki boyutlu sistemler için kullanılan ağaç veri yapısı dörtlü ağaç (quadtree), üç boyutlu sistemler için kullanılan ağaç veri yapısı ise sekizli ağaç (octree) olarak adlandırılır. Ağaç yapısında kullanılan terimler açıklanırken anlatım kolaylığı açısından iki boyutlu problem ele alınacaktır. Dörtlü ağaç veri yapısında kök düğüm (root node) tüm sistemi içine alacak büyüklükte olan kareyi (hücre) temsil etmektedir. Bir düğüme karşılık gelen kare dört eşit parçaya bölünerek dört alt-düğüm (çocuk düğüm) ve dört eşit alt-kare elde edilir. Dörde bölünen kareye ebeveyn, bölümlenmiş olan bu dört küçük kareye ise çocuk kare denir. En alt seviyedeki düğümler yaprak düğüm olarak adlandırılır. Bir cisim aynı sevide bulunan karelerden sadece bir tanesinin içinde bulunabilir. Hiyerarşik olarak bölümleme işleminde kareler birbirlerini kesmezler. Aynı seviyede bulunan bir kare ile herhangi bir kenarı ortak olan kareye komşu kare denir.

2.4 AĞAÇ YAPISININ OLUŞTURULMASI :

Tüm sistemi içine alan kareye cisimler teker teker yerleştirerek ağaç veri yapısı oluşturulmaktır. Bu yöntemde kareye cisim yerleştirilir ve işlem sonunda karede en fazla k tane cisim ($k \geq 1$ gerçeklemeye bağlı olarak seçilebilir) kalma şartı kontrol edilir. Eğer k+1 tane cisim varsa şart sağlanana kadar bu kare dört eşit kareye bölünür ve cisimler uygun karelere yerleştirilir. Bölümleme sonunda oluşan kareler ağaç veri yapısında kök düğümden başlanarak saklanır. İşlem sonunda karede cisim yoksa bu kare işleme alınmaz ve bu kareye karşılık ağaç veri yapısında bir düğüm bulunmaz. Böylece cisimler hiyerarşik olarak gruplanmış ve etkileşim için hazırlanmış olur.

Düzenli dağılmış bir sistemde her bir bölümleme sonucunda 4 tane çocuk kare oluşacaktır ve en alt seviyedeki karelerde en fazla k tane cisim bulunacaktır. Toplam cisim sayısı N, en alt seviyeye h dersek bu seviyede toplam 4^h tane kare olacaktır. Bu durumda h yaklaşık olarak logN'dir. Sisteme bir cisim ekleyebilmek için ağaç üzerinde en fazla h seviye aşağı inilir. Bu durumda bir cismin ağaç üzerindeki yerini belirleyebilmek için O(LogN) işlem yapılması gerekir. Tüm cisimlerin ağaç üzerine yerleştirilmesi O(NlogN) işlemde tamamlanacaktır. Düzensiz bir dağılım ağacın seviyesinin dengesiz olarak artmasına ve O(NlogN) 'lik işlem zamanının $O(N^2)$ 'ye yükselmesine neden olmaktadır.

2.5 BARNES-HUT ALGORİTMASI

Barnes-Hut algoritması 1986 yılında yayınlanmıştır. Barnes-Hut'un algoritmasının Appel'e göre sağladığı avantajlar ağaç yapısındaki karışıklıkların engellemesi ve hata için üst sınır verip ortalama bir hata hesabı yapmayı sağlamasıdır. Bir kenar uzunluğu r olan bir karenin içindeki cisimler gruplanarak bu karenin kütle merkezinde tek bir cisim varmış gibi kabul edilir.

Kütle merkezindeki bu sanal cisim ile yeterince uzakta olan cisimlerle etkileşim yapılır. Buna cisim-küme etkileşimi denir. Cisim ile karenin kütle merkezi arasındaki uzaklık (d), karenin bir kenarının uzunluğundan büyük olmalıdır ve bu r/d oranını bir Θ parametresi ile kontrol edilmektedir. Parametre 0 ve 1 arasındadır. Bu parametre algoritma sonundaki doğruluk oranını ve çalışma zamanını etkilemektedir.

Ağaç yapısı oluşturulduktan sonra ağaç yapısı üzerinde yapraklardan başlanarak en üstteki düğümlere çıkılır ve kümelerin kütle merkezleri ve toplam kütlesi hesaplanır. Örnek olarak 4 çocuk düğümü (kümesi) bulunan bir küme için 4 çocuk kümenin toplam kütleleri ve kütle merkezleri hesaplandıktan sonra bu 4 çocuğun kütle merkezleri ebeveyn kümesinin küme merkezine kaydırılıp kütleler toplanarak toplam kütle elde edilir. Böylece ağacın her bir seviyesindeki kütle merkezlerinin kolayca hesaplanması sağlanır.

Kütle merkezleri hesaplandıktan sonra kuvvet hesabına geçilir. Her bir cisim için ağaç üzerinde kök düğümden başlanıp yaprak düğümlere inilir ve her bir düğüm ile cismin için Θ parametresi kontrol edilir. Eğer bu şart sağlanıyorsa cisim ile bu düğümün cisim-küme etkileşimi yapılır ve bu düğümün alt düğümlerine inilmez. Eğer şart sağlanmıyorsa bu düğümün alt düğümlerinin her birisi için tekrar bu şart kontrol edilir ve işleme devam edilir. Eğer yaprak düğüme gelindiyse bu durumda cisim-cisim etkileşimi yapılır.

Ağacın oluşturulması O(NlogN) zamanda ve kuvvet hesabı O(NlogN) zamanda yapılmaktadır. Bu durumda toplam çalışma zamanı O(NlogN) olacaktır. Kuvvetler hesaplandıktan sonra cisimler yeni konumlarına kaydırılır ve bir sonraki zaman aralığına geçilir. Ağaç tekrar oluşturulur ve tüm hesaplamalar yeniden yapılır.

2.6 HIZLI ÇOK-KUTUP ALGORİTMASI

Hızlı Çok-kutup Algoritması (Fast Mulipole Method, HÇA) Leslie Greengard ve Vladimir Rokhlin tarafından geliştirilmiştir[2][4]. 20. yüzyılın en popüler 10 algoritması arasında gösterilmektedir. Hızlı Çok-kutup Algoritması Barnes-Hut algoritmasında olduğu gibi cisim-küme etkileşimi yapmak yerine küme-küme etkileşimi yapar. Küme-küme etkileşimi yaparken Taylor serisi açılımı yardımıyla potansiyelleri hesaplayarak kuvvetleri yaklaşık olarak hesaplayıp işlem zamanını O(N) adıma indirilmiştir. Bu yakınsamadan dolayı oluşacak hata seri açılımlarında belli sayıda terim alınarak sınırlandırılabilmektedir.

Hızlı Çok-kutup algoritması iki boyutlu problemler için çok yüksek doğruluk oranında hesaplama yapılmasını sağlar. Üç boyutlu problemlerde ise bu doğruluk oranını sağlayamadığı gibi çalışma zamanı da oldukça artmaktadır. Üç boyutlu problemlerde pek çok doğruluk oranı için etkili bir şekilde çalışacak yeni bir Hızlı Çok-Kutup algoritması geliştirilmiştir[5]. Bu yeni algoritma düzensiz dağılımlar için adaptif çözümde sunar. Fakat bu algoritmanın gerçeklenmesi ise oldukça zordur.

Hızlı Çok-kutup Algoritmasının Barnes-Hut Algoritmasından farklı yönleri:

1 Barnes-Hut Algoritmasındaki gibi her cisim üzerindeki kuvvetler değil potansiyeller hesaplanır. Vektörel bir büyüklük olan kuvvet yerine skaler bir büyüklük olan potansiyel ile hesaplamalar işlemleri kolaylaştırmaktadır.
2 Küme merkezi ve kütle bilgileri yerine seri açılımlar kullanılır. Bu açılımda daha fazla terim alarak daha hassas hesaplamalar yapılabilmektedir. Daha fazla terim alma işlem zamanını ve bellek kullanımını artıracaktır.
3 Kümeler Barnes-Hut Algoritmasında olduğu gibi Θ parametresine göre değil, en fazla 27 (iki boyut için) tane küme ile etkileşime girmektedir.

Bir daire içinde bulunan cisimlerin, bu dairenin dışında (yeterince uzakta) bulunan cisimler için potansiyel hesabını tek tek yapmak yerine çok-kutup açılımı yardımıyla elde etmek mümkündür. Bir dairenin dışında yeterince uzakta bulunan tüm cisimlerin dairenin merkezindeki potansiyeli hesaplamayı sağlayacak seri açılıma yerel açılım denir. Yerel açılımların hesaplanması için önce çok-kutup açılımlarının hesaplanmış olması gerekir. Algoritmada kullanılan seri açılımlar sonsuz sayıda terim içermektedir. Fakat açılımı bilgisayar yardımıyla hesaplayabilmek için p tane terim alınır. p terimden sonsuza kadar olan terimler toplamı hatayı göstermektedir. p terimi büyük seçilerek daha doğru sonuçlar elde edilebilir.

Algoritma üç adımdan oluşmaktadır. Birinci adımda ağaç yapısını oluşturur ve bu işlem sonunda her bir yaprak düğümde en fazla k tane cisim bulunur. k değeri cisim sayısına göre belirlenebilir. İkinci adımda aşağıdan yukarıya çıkılarak çok-kutup açılımlarının hesaplanması sağlanır.Bu hesaplama Barnes-Hut algoritmasındaki kütle merkezi hesaplanmasına benzemektedir. Çok-kutup açılımını Φ ile gösterelim. Önce tüm yaprak düğümlerdeki Φ hesaplanır. Sonra bu düğümlerin ebeveyn düğümleri için çok-kutup açılımı, çocukların çok-kutup açılımlarının merkezinin ebeveyn düğümün merkezine kaydırılmasıyla hesaplanır ve ebeveynin Φ değerine ilave edilir. Bu işleme kök düğüme kadar devam edilir.

Son adımda ise yukarıdan aşağıya inilerek yerel açılımlar hesaplanır. Yerel açılımı Ψ ile gösterelim. Yerel açılımın hesaplanması için önce düğümlerin etkileşim listeleri belirlenir. Bir düğümün etkileşim listesi bu düğümün ebeveyn düğümünün yakın

komşularının çocuklarından bu düğümle ortak bir kenarı bulunmayan (yakın komşusu olmayan) düğümlerdir. Yerel açılımın hesaplanması için karenin etkileşim listesinde bulunan karelerdeki Φ çok-kutup açılımları, Ψ yerel açılımına çevrilir. Bir seviyedeki tüm düğümler için yerel açılımlar hesaplanır ve bu yerel açılımlar çocuk düğümlerin merkezlerine kaydırılır.

Bir kare için yerel açılım hesabını inceleyelim.A karesinin yerel açılımını hesapladığımızı düşünelim. Bu A karesinin ebeveyni olan B karesinin yerel açılımı A'nın merkezine kaydırılır. Bu yerel açılım B karesinin yeterince uzakta bulunan tüm cisimlerin B karesine olan potansiyel etkisinin hesaplanması sağlamaktadır. Fakat A karesinin etkileşim listesinde B karesinin yerel açılımda olmayan kareler vardır. Bu etkileşim listesindeki tüm karelerdeki çok-kutup açılımlar A karesinin yerel açılımı olarak çevrilir. Tüm etkileşim listesindeki çok-kutuplar çevrildikten sonra A karesinin yerel açılımı varsa A karesinin tüm alt karelerinin merkezlerine kaydırılır. Her bir alt kare için etkileşim listesindeki çok-kutup açılımları yerel açılıma çevrilerek işleme devam edilir.

En alt seviyede bulunan yaprak düğümlere gelindiğinde elimizde yakın komşular dışındaki cisimlerin için potansiyel hesabını yapmamızı sağlayacak yerel açılım vardır. Bu yerel açılım karenin içindeki her bir cismin konumlarına kaydırılır. Bu karenin içindeki cisimlerin karenin yakın komşuları ile olan hesaplamaları herhangi bir açılım kullanılmadan doğrudan hesaplanır ve yerel açılımdan elde edilen potansiyel ile toplanır. Potansiyelden kuvvete, oradan da konumlara geçilir.

4 DENEYLER

Bu bildiride cisim-cisim, Barnes-hut ve Hızlı çok-kutup algoritmaları C++ dili kullanılarak gerçeklenmiş ve yapraklardaki cisim sayısının değişiminin çalışma zamanı ve doğruluk oranlarına etkisi incelenmiştir. Deneylerde kullanılan cisimlerin konumları rastgele sayı üreticisi yardımıyla 0 ve 1 arasında üretilmiştir. Bir adım sonraki kuvvetler hesaplanmış ve algoritmaların yaptıkları hatalar karşılaştırılmıştır. Kuvvetler hesaplandıktan sonra konumların hesaplanması için Leapfrog integrasyon yöntemi kullanılmıştır. Önce cisim-cisim hesaplaması yapılarak bir adım sonraki kuvvetler bulunmuş ve bu elde edilen kuvvetler doğru kabul edilip, Barnes-Hut ve Hızlı çok-kutup algoritmalarının sonunda elde edilen kuvvetler ile farkı alınarak ortalama hata hesaplanmıştır. N tane cisim için ortalama hata hesaplanırken :

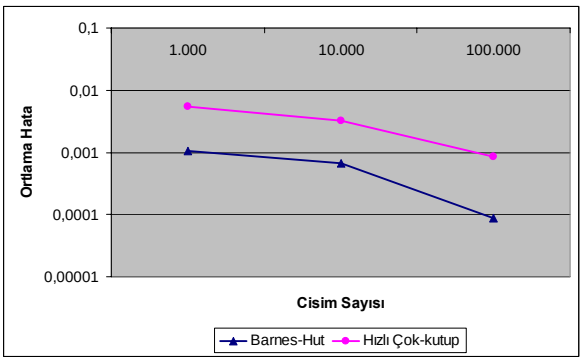
$$\text{Ort\_Hata} = \left[\frac{1}{N} \sum_{i=1}^{N} \left(\frac{f_{hiyerarşik} - f_{cisim-cisim}}{f_{cisim-cisim}} \right)^2 \right]^{1/2}$$

formülü kullanılmıştır. Burada Barnes-hut veya Hızlı çok-kutup ile hesaplanan kuvvetler $f_{hiyerarşik}$, cisim-cisim ile hesaplanan kuvvetler $f_{cisim-cisim}$ şeklinde gösterilmiştir.

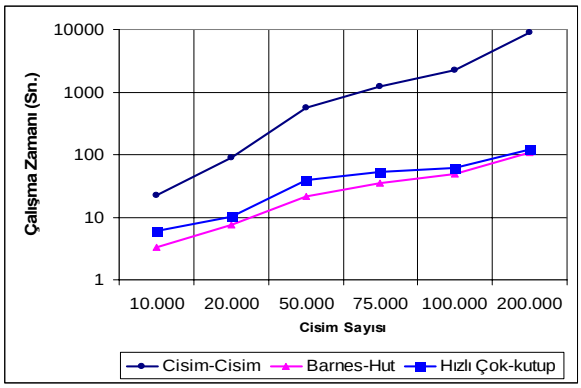
Barnes-Hut ve Hızlı çok-kutup için aynı ağaç veri yapısı kullanılmıştır. Her iki algoritma içinde ortak bir yapı kullanılmaya çalışılmış fakat algoritmalarının birbirinden farklı olan kısımları için ayrı fonksiyonlar yazılmıştır. Bu bildiride gerçeklenen algoritmalar orijinal makalelerine [1][2] bağlı kalınarak gerçeklenmiştir. Hızlı çok kutup algoritmasındaki çok-kutup açılımlarının ve yerel açılımlarının hesaplanmasını sağlayacak matematiksel hesaplamalar için kodlar [6]'dan alınmıştır.

Her iki algoritmada ağaç veri yapısı her bir adımda yeniden oluşturulmaktadır. Her bir adımda new komutu ile bellekten yeni bir alan tahsisi yerine önceden bellekten tahsis edilmiş bir alan kürsor(cursor) mantığı ile kullanılmıştır. Düzenli dağılmış bir sistem için kullanılan düğüm(Node) sayısı genelde 3N/2 olarak gözlenmiştir. Fakat bu sayı yapraklardaki cisim sayısının değişmesiyle azalmaktadır.

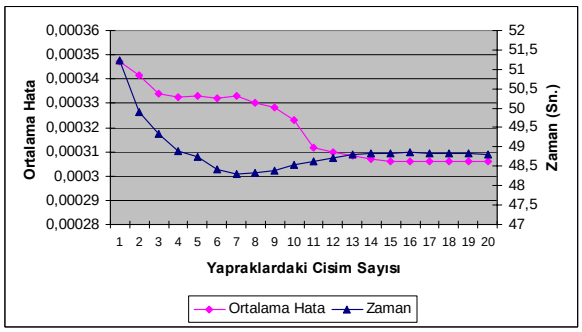
Deneyler üzerinde Windows XP işletim sistemi yüklü, Pentium 4 3.2 işlemci ve 1 GB bellek bulunan bilgisayarda yapılmıştır. Barnes-Hut algoritması teta değeri 5 ve Hızlı Çok-kutup algoritmasındaki çok-kutup açılımındaki p terimi 3 alınarak çeşitli sayıda cisim için gerçeklemeler çalıştırılmış hata oranları Şekil 2 'de, çalışma zamanları Şekil 3 'de gösterilmiştir.



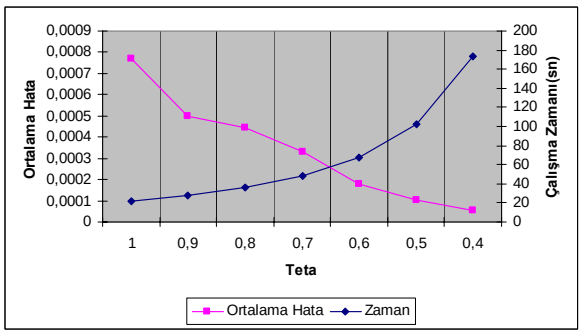
Şekil 2 – Cisim-cisim , Barnes-Hut ve Hızlı Çok-kutup algoritmalarının cisim sayısı ve ortalama hata değişimi



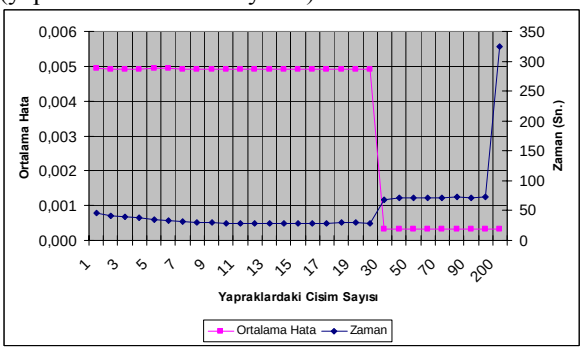
Şekil 3 – Cisim-cisim , Barnes-Hut ve Hızlı Çok-kutup algoritmalarının cisim sayısı ve çalışma zamanı değişimi



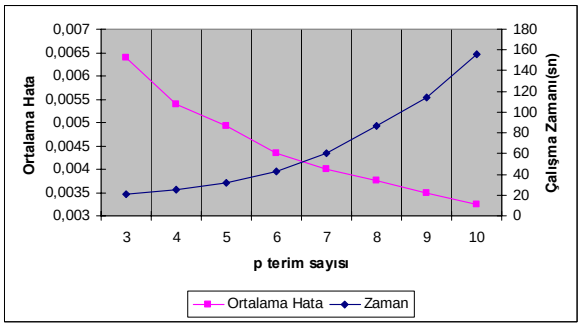
Şekil 4 – 100.000 cisim için Yaprak düğümlerde bulunan cisim sayısının artırılması ile ilgili Barnes-hut algoritmasındaki hatanın değişimi (Teta 0,7)



Şekil 5 – 100.000 cisim için Barnes-Hut algoritmasındaki teta ile ortalama hata değişimi (yapraklardaki cisim sayısı 7)



Şekil 6 – 100.000 cisim için Hızlı çok-kutup algoritmasındaki yapraklardaki cisim sayısının ortalama hataya etkisi (p=5)



Şekil 7 – 100.000 cisim için Hızlı çok kutup algoritmasındaki p terim sayısı ile ortalama hata değişimi (yapraklardaki cisim sayısı 7)

5 SONUÇ

Yapılan deneyler sonunda düşük doğruluk oranında Barnes-hut algoritması hızlı çok-kutup algoritmasından daha hızlı çalışmaktadır. Fakat yüksek doğruluk oranı istendiğinde Hızlı çok-kutup algoritması $N>10^4$ olmak şartıyla Barnes-Hut algoritmasından daha hızlı çalışmaktadır. Yapraklardaki cisim sayısı artıkça doğruluk oranı ve çalışma süresi artmaktadır. Optimum bir değerden bahsetmek mümkün olmamıştır. Bu sayı cisim sayısına bağlı olarak değişmektedir.

KAYNAKLAR

[1] J. Barnes and P. Hut, A hierarchical O(N log N) force-calculation algorithm, Nature 324, 446-449 (1986).

[2] L. Greengard and V. Rokhlin, A Fast Algorithm for Particle Simulations**,** JOURNAL OF COMPUTATIONAL PHYSICS Vol. 73, Number 2, December (1987), pages 325–348.

[3] A. W. Appel. An Efficient Program for Many-body Simulation. Siam. J. Sci. Stat. Comput. 6, 85–103 (1985).

[4] V. Rokhlin, Rapid Solution of Integral Equations of Classical Potential Theory, J. Comp. Phys. v. 60

[5] H. Cheng, L. Greengard and V. Rokhlin, A fast adaptive multipole algorithm in three dimensions, Journal of Computational Physics 155,468-498(1999)

[6] William T. Rankin and John A. Board, Jr., A Portable Distributed Implementation of the Parallel Multipole Tree Algorithm, Proceedings of the Fourth IEEE International Symposium on High Performance Distributed Computing - HPDC '95, Page: 17