EGE ÜNİVERSİTESİ FEN BİLİMLERİ ENSTİTÜSÜ

(YÜKSEK LİSANS TEZİ)

AYRIŞTIRMA TABANLI KARARLI DİZİN BİRLEŞTİRME ALGORİTMALARI ÜZERİNE BİR ARAŞTIRMA

İlker KOCABAŞ

Uluslararası Bilgisayar Anabilim Dalı

Bilim Dalı Kodu : 619.03.03

Sunuş Tarihi : 16.03.2005

Tez Danışmanı : Prof. Dr. Mehmet Emin Dalkılıç

BORNOVA - İZMİR

İlker KOCABAŞ tarafından YÜKSEK LİSANS TEZİ olarak sunulan "**Ayrıştırma Tabanlı Kararlı Dizin Birleştirme Algoritmaları Üzerine Bir Araştırma**" başlıklı bu çalışma E.Ü. Fen Bilimleri Enstitüsü Eğitim ve Öğretim Yönergesi'nin ilgili hükümleri uyarınca tarafımızdan değerlendirilerek savunmaya değer bulunmuş ve 16.03.2005 tarihinde yapılan tez savunma sınavında aday oybirliği/oyçokluğu ile başarılı bulunmuştur.

Jüri Üyeleri:		İmza:
Jüri Başkanı	:	.....
Raportör Üye:	.....	.....
Üye	:	.....

ÖZET

AYRIŞTIRMA TABANLI KARARLI DİZİN BİRLEŞTİRME ALGORİTMALARI ÜZERİNE BİR ARAŞTIRMA

KOCABAŞ, İlker

Yüksek Lisans Tezi, Uluslararası Bilgisayar Enstitüsü

Tez Yöneticisi: Prof. Dr. Mehmet Emin DALKILIÇ

Mart 2005, 98 sayfa

Bir çok uygulama özel sıralama işlemleri için birleştirme (merging) algoritmalarına ihtiyaç duymaktadır. Kararlı (stable) ve/veya yerinde (in-place) birleştirme algoritmaları, ayrıştırma (decomposition) tabanlı bir yol izlenerek daha kolay ve etkin bir biçimde gerçekleştirilebilmektedir.

Bu tez projesinde; özyinelemeli ve kararlı bir ayrıştırma tabanlı birleştirme algoritması (ATBA) iyileştirilmiş ve iyileştirilmiş algoritmanın özyinelemeden kurtarılarak yerinde hale getirilmiş uyarlaması sunulmuştur. Bu algoritmalara ek olarak, literatürde mevcut ATBA'lar da gerçekleştirilmiş ve daha sonra birleştirme ve sıralama uygulamalarında çalışma süreleri testleri yapılmıştır. Test sonuçlarına göre, her iki uygulamada da burada sunulan algoritmaların performanslarının kıyaslanan algoritmalara göre çok daha iyi olduğu gözlenmiştir. Yine bu tezde, seri ortamlarda çalışan iyileştirilmiş, kararlı ve yerinde birleştirme algoritmasının ortak bellekli paralel mimariler için düzenlenmiş iki uyarlaması sunulmuştur. Paralel algoritmalar ortak bellek benzetimi yapılmış paralel ortama uygun olarak gerçekleştirilmiştir. Seri ve paralel test ortamında ölçülen çalışma sürelerine göre algoritmaların hızlanmaları ve verimlilikleri incelenmiştir.

Tüm algoritmalar ve test programları Linux platformları için C programlama diliyle kodlanmıştır. Ayrıca paralel çalıştırma ortamı - Linux küme platformu- MPI kütüphanesi kullanılarak sağlanmıştır.

Anahtar sözcükler: ayrıştırma tabanlı, özyinelemeli, kararlı, yerinde, ortak bellekli

ABSTRACT

A STUDY ON DECOMPOSITION BASED STABLE MERGING ALGORITHMS

KOCABAŞ, İlker

MSc. in International Computer Institute

Supervisor: Prof. Dr. Mehmet Emin DALKILIÇ

March 2005, 98 page

For special sorting processes, wide range of applications need merging algorithms. Stable and /or in-place merging algorithms may be implemented in much more easy and effective manner following decomposition based approach.

In this thesis, a recursive and stable decomposition based merging algorithm is improved and then an in-place version of the improved algorithm, freed of recursion, is presented. In addition to these algorithms, decomposition based merging algorithms which is available in literature are implemented and run time tests are carried out in merging and sorting applications. According to test results, it is observed that presented algorithms performances are significantly higher than that of the compared algorithms in both applications. Also in this thesis, two versions of the improved, in-place and stable algorithm adapted for the shared memory parallel architectures are presented. The parallel algorithms are implemented for the parallel environment which is simulated shared memory. Speed-ups and efficiencies of algorithms with respect to their run times measured both in serial and parallel environments are investigated.

All algorithms and test programs are coded in C programming language for Linux platforms. Furthermore, parallel environment –Linux cluster platform- is provided by using MPI library.

Keywords: decomposition based, recursive, stable, in-place, shared memory

TEŞEKKÜR

Öncelikle bu tez konusu üzerinde bana çalışma imkanı sunan tez danışmanım Prof. Dr. M. Emin Dalkılıç'a çalışma süresince deneyimi, bilgisi ve önerileriyle araştırma ve geliştirmeyi yönlendirmesi ve sağladığı kaynaklarla destek olmasından dolayı teşekkürü bir borç bilirim.

Bu çalışmamı daima yanımda olan ve manevi desteklerini esirgemeyen aileme; annem, babam ve kardeşime adıyorum.

İÇİNDEKİLER

Sayfa

İÇİNDEKİLER (devam)

Sayfa

ŞEKİLLER DİZİNİ

Şekil Sayfa

ŞEKİLLER DİZİNİ (devam)

Şekil Sayfa

ÇİZELGELER DİZİNİ

Çizelge Sayfa

KISALTMALAR

AT1	: Ayrıştırma Tabanlı Birinci Algoritma (Dudzinsky ve Dydek'in Algoritması)
AT2	: Ayrıştırma Tabanlı İkinci Algoritma (Dalkılıç'ın Algoritması)
ATBA	: Ayrıştırma Tabanlı Birleştirme Algoritması
ATD1	: Ayrıştırma Tabanlı Değiştirilmiş Birinci Algoritma (Dvorak ve Durian'ın Özyinelemeli Algoritması)
ATD2	: Ayrıştırma Tabanlı Değiştirilmiş İkinci Algoritma (Geliştirilen AT2 Tabanlı Özyinelemeli Algoritma)
ATY1	: Ayrıştırma Tabanlı Yerinde Birinci Algoritma (Dvorak ve Durian'ın Yerinde Algoritması)
ATY2	: Ayrıştırma Tabanlı Yerinde İkinci Algoritma (Geliştirilen AT2 Tabanlı Yerinde Algoritma)
ATP1	: Ayrıştırma Tabanlı Paralel Birinci Algoritma (Geliştirilen ATD1 Ayrıştırma Süreci Kullanan Paralel Algoritma)
ATP2	: Ayrıştırma Tabanlı Paralel İkinci Algoritma (Geliştirilen ATD2 Ayrıştırma Süreci Kullanan Paralel Algoritma)
BBA	: Basit Birleştirme Algoritması
BU	: Buttom-Up (Aşağıdan-Yukarıya)
CREW	: Concurrent Read Exclusive Write (Koşutzamanlı Okuma ve Dışlayan Yazma)
MPI	: Message Passing Interface (İleti Geçirme Arabirimi)
PRNG	: Pseudo Random Number Generator (Sözde Rastgele Sayı Üreteci)
SBA	: Standart Birleştirme Algoritması
TD	: Top-Down (Yukarıdan-Aşağıya)

1 GİRİŞ

Birleştirme işlemi, birçok uygulama tarafından ihtiyaç duyulmasından dolayı, günümüzde üzerinde çok çalışılan bir konu olarak karşımıza çıkmaktadır. Bir birleştirme algoritması kolayca iyi bir sıralama algoritmasına dönüştürülme özelliğine sahiptir. Sıralama problemi de bilgisayar bilimlerinin en temel problemlerinden birisi, belki de en önemlisidir.

Birleştirme algoritmaları sıralama algoritmalarından BirleştirSırala'nın (Mergesort) birleştirme evresinde uygulanır. Bu yüzden çalıştırılan birleştirme algoritmasının verimliliği, etkinliği ve barındırdığı özellikleri doğrudan bu sıralama algoritmasının performansını etkiler. Knuth tarafından ortaya konulan problemin:

> *"En kötü veya ortalama durumda $O(N^2)$ çalışma süresinin altında minimum belleğe sahip kararlı bir sıralama algoritması var mı?" (Knuth, 1973)*

BirleştirSırala ve birleştirme evresiyle çözülmesi birleştirme algoritmalarının önemini artırmıştır. Günümüzde bu sorunun cevabı olan başka bir sıralama algoritması yoktur.

Birleştirme ve sıralama algoritmalarının verimlilik, etkinlik ve özellik kriterleri aynıdır. Algoritmaların verimliliği bir veri kümesi üzerindeki çalışma süresi ve bellek olarak kullandığı kaynakların miktarı bakımından ölçülebilir. Bu amaçla verimliliğin matematiksel olarak analizi elemanlarındaki taşıma, karşılaştırma sayılarının ve bellek karmaşıklıklarının hesaplanmasıyla gerçekleşir. Ancak gerçekleşen diğer işlemlerin hesaba katılmaması, çalıştığı platforma uygun olmaması, zayıf önbellek (cache) kullanımı ile kaçırma oranının (miss rate) yüksek olması (Li Xio et al., 2000) gibi sebeplerden dolayı sınırlı matematiksel verimlilik analizleri doğru bir karşılaştırma aracı olmayabilir. Bu yüzden diğer bir verimlilik ölçüsü de algoritmaların aynı platformda ve değişik

veri kümeleri üzerinde çalıştırılmasıyla elde edilen gerçek çalışma süreleridir. Algoritmaların etkinliği ise anlaşılabilir ve kolay bir biçimde gerçeklenebilir olmasıyla ölçülür. Geniş bir uygulanma sahası olan bir işlem için geliştirilen bir algoritmanın yaygın olarak kullanılabilmesi verimliliği kadar önemlidir. Algoritmaların özellikleri kullanılacağı uygulamanın ihtiyaçlarını karşılayıp karşılayamayacağını gösteren bir ölçüdür. Üzerinde çalışabileceği giriş elemanlarının yapısı, işlem sayısı arasıdaki ilişkileri, bellekte tutulma şekilleri, işlem sonunda birbirlerine göre bağıl pozisyonlarının durumu gibi özelliklerden dolayı farklılıklar gösterirler.

Giriş elemanlarının yoğun kütleli olduğu uygulamalarda -büyük veri tabanları gibi- birleştirme işlemi ek bellek kullanımının sabit olmadığı durumlarda gereksiz bellek kullanımına yol açmaktadır. Birincil amacın bellek alanı kullanımının minimum olması hedeflenen bu tür uygulamalarda, birleştirme işlemenin yerinde olarak yapılması gerekmektedir. Yine bazı uygulamalarda giriş kümesindeki elemanların bağıl pozisyonlarının birleştirme işleminden sonra korunması istenmektedir. Mesela dağıtık veritabanlarında alfabetik olarak sıralı bulunan öğrenci bilgileri mezuniyet tarihlerine göre sıralandıktan sonra aynı mezuniyet tarihindeki öğrencilerin alfabetik sırasının korunması istenebilir. Bu ihtiyaçlara sahip uygulamalarda makul ölçülerde çalışma hızı yeterli olabilmektedir.

Bu tez çalışmasında ele alınan problem tamsayı anahtar değerlerine göre sıralanmış iki dizinin birleştirilmesidir. Birincil hedef hem kararlı hem de yerinde özelliğe sahip etkin ve makul verimlilikte bir birleştirme algoritması geliştirmektir. Bu çözüme ulaşmakta izlenecek yol bölümleme temeline dayanan, gerçeklenmesi basit ayrıştırma tabanlı yaklaşımdır. Ayrıştırma tabanlı olmayan yaklaşımla geliştirilen optimal zaman karmaşıklığına sahip algoritmaların etkinlikleri, kullandıkları

karmaşık teknikler ve fazla sayıdaki kontrollerden dolayı iyi değildir. İkincil hedef ise kararlı özelliğe sahip, yeni veya önceden geliştirilmiş ayrıştırma tabanlı algoritmaların incelenerek birbirleriyle kıyaslanmasıdır.

Bu tez çalışmasında ilk olarak biri sabit miktarda bellek kullanan olmak üzere kararlı özelliğe sahip iki farklı ayrıştırma tabanlı birleştirme algoritması geliştirilmiştir. Daha sonra geliştirilen algoritmalardan kararlı ve yerinde olan algoritma ortak bellekli çok-işlemcili sistemler için uygun hale getirilerek paralel çalışan kararlı ve yerinde iki versiyonu geliştirilmiştir. Paralel algoritmaların temel farklılığı giriş probleminin belirli parçalarının –alt problemlerin- işlemcilere dağıtılmasını sağlayan iş paylaşım sürecinde izlenen yoldur. Tüm geliştirilen algoritmalar C diliyle Unix/Linux platformlarına uygun olarak gerçeklenmiştir. Paralel algoritmaların gerçeklenmesi sırasında ek olarak MPI kütüphanesi kullanılmıştır.

Seri olarak çalışan ayrıştırma tabanlı birleştirme algoritmalarının çalışma sürelerinin kıyaslanabilmesi için literatürde bulunan kararlı ayrıştırma tabanlı algoritmalar ile optimal zaman karmaşıklığına sahip yerinde olmayan standart birleştirme algoritması kodlanmıştır. Gerçekleştirimlerin testleri anahtar değer kümesi, dizin uzunluğu ve baştaki blok uzunluğunun dizin uzunluğuna oranı gibi parametrelerin değişik değerleri için 2 GHz işlemcili Linux bilgisayarlarında yapılmıştır. Bölüm 4.1'de bu test sonuçları verilerek değerlendirilmiştir. Geliştirilen iyileştirilmiş özyinelemeli kararlı algoritmanın performansı rastgele sıralı giriş dizileri için en iyi sonucu vermiştir. Bu algoritmayı temel alarak geliştirilen yerinde uyarlaması ise aynı veri kümesi üzerinde –rastgele üretilen giriş problemi- Dvorak ve Durian'ın yerinde algoritmasından daha hızlı çalışmaktadır.

İkinci bir kıyaslama da geliştirilen algoritmaların

BirleştirSırala'daki (MergeSort) etkileri üzerine yapılmıştır. Bu algoritmalarla birlikte yerinde ve kararlı özelliklere sahip Dvorak ve Durian'ın ayrıştırma tabanlı algoritmalarının (Dvorak and Durian, 1988) birleştirme evresinde kullanıldığı BirleştirSırala algoritmaları ile performansı yüksek -iyileştirilmiş-, özyinelemeli olarak geliştirilmiş bir BirleştirSırala sıralama algoritması (Sedgewick, 1998) C diliyle gerçekleştirilmiştir. Farklı uzunluktaki dizinler için değişik parametrelerle 2 GHz işlemcili Linux bilgisayarlarında testler gerçekleştirilmiştir. Bölüm 4.2'de bu test sonuçları verilerek değerlendirilmiştir. Ayrıştırma tabanlı algoritmaların birleştirme gerçekleştirimlerindeki göreceli performanslarının sıralama uygulamalarında aynı biçimde devam ettiği gözlenmiştir.

Son olarak ise ortak bellekli paralel mimariler için uygun hale getirilen kararlı ve yerinde algoritmalar, MPI kütüphanesi kullanılarak ortak bellek benzetimine uygun biçimde gerçekleştirilmiştir. Bu benzetim ortamında çalışma süreleri farklı dizin uzunluklarında ve birleştirilecek dizilerin uzunluklarının birbirlerine oranlarının farklı değerleri için yapılmıştır. Sekiz bilgisayarlı (aynı işlemci gücüne ve bellek erişim süresine sahip) Linux kümesinden oluşan çalıştırma ortamında gerçekleştirimler 2, 4 ve 8 işlemci ile test edilmiştir. Hızlanma ve verim hesaplarında her bir algoritma için temel alındıkları seri algoritmanın çalışma süresi kullanılmıştır. Bölüm 4.3'de bu test sonuçları incelenip değerlendirilmiştir.

Tezin bundan sonraki organizasyonu şu şekildedir: İkinci bölümde, tezin hazırlanmasında gerekli olan alt yapı bilgileri, kullanılan teknikler ve daha önce bu konuda gerçekleştirilmiş olan çalışmalar yer almaktadır. Üçüncü bölümde, geliştirilen seri ve paralel algoritmalar hakkında detaylı bilgi verilmektedir. Dördüncü bölüm, seri birleştirme algoritmalarının, bu algoritmaların kullanıldığı BirleştirSırala (MergeSort) algoritmalarının ve

paralel birleştirme algoritmalarının test sonuçlarını içermektedir. Ayrıca geliştirilen seri algoritmaları kullanan BirleştirSırala sıralama algoritmalarının gerçeklenmesi ve paralel algoritmaların benzetim ortamına uygun hale getirilmesiyle ile ilgili detaylar da bu bölümde yer almaktadır. Test sonuçları ile ilgili incelemeler ve değerlendirmeler yine bu bölümde yer almaktadır. Beşinci bölüm ise sonuç bölümüdür. Tez çalışması sonucunda ortaya konan algoritmalar ve ileriye yönelik yapılabilecek çalışmalar hakkında bilgi verilmiştir. Bu bölümü kaynaklar dizini ve ekler takip etmektedir.

2 İLGİLİ ÇALIŞMALAR

Bu bölümde, daha önce bu konuda yapılmış olan çalışmalar ve kullandıkları yöntemler açıklanmaktadır.

2.1 Genel Bilgiler

Birleştirme işlemi Knuth tarafından aşağıdaki gibi tanımlanmıştır:

> *"İki veya daha fazla anahtar elemanlarına göre sıralı dizinin sıralı tek dizi haline getirilmesidir" (Knuth, 1973).*

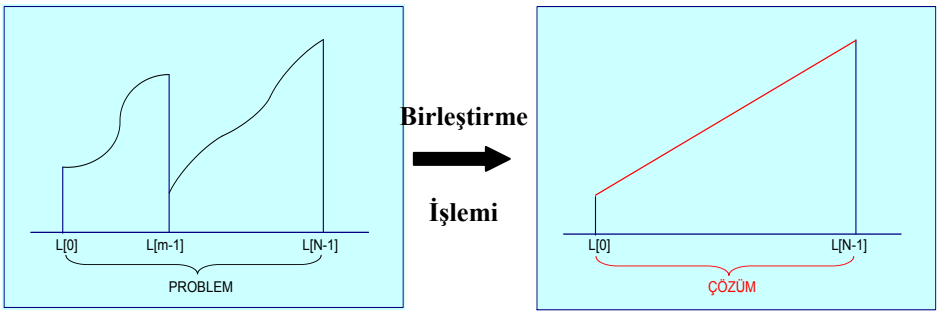
Birleştirme işlemini gerçekleştiren algoritmalar farklı özellikler gösterirler. İhtiyaç duyuldukları değişik uygulamalar tarafından uygun olan birleştirme algoritmasının seçilmesini bu özellikler sağlarlar. Algoritmaların özellikleri sıralama algoritmalarında olduğu gibi birleştirilecek dizilerin veri yapısı, birleştirecek dizilerin bellekte tutuldukları alan, dizilerdeki elemanların anahtar değerlerinin işlem sonrasında birbirleriyle ve işlem miktarıyla ilişkisi, ek bellek kullanım miktarı gibi kriterlerdeki ayrımlarından ortaya çıkar (Sedgewick, 1998).

Bir birleştirme algoritması dizin, ağaç veya bağlı liste gibi veri yapılarından birisine sahip diziler üzerinde çalışır. Ana bellekte, *içsel birleştirme (internal merge)*, veya teyp gibi dış cihazlarda, *dışsal birleştirme (external merge)*, tutulan dizilerde uygulanabilir. Algoritma, dizilerin anahtar eleman değerlerindeki değişiklik işlem sayısını etkiliyor ise *uyarlamalı (adaptive)*, etkilemiyor ise *uyarlamasız* diye adlandırılır. Uyarlamalı birleştirme/sıralama algoritmaları bir sıra içinde varolan düzenin yararını göz önünde bulundurduklarından dolayı özellikle giriş elemanlarının yaklaşık olarak sıralı ortaya çıkmasının sıklıkla görüldüğü uygulamalar tarafından kullanılmaktadır (Estivill-Castro and Wood, 1992). Birleştirme işlemi sonrası aynı anahtar değerlerine sahip dizi elemanlarının bağıl pozisyonlarının korunup/korunmamasına göre

kararlı/kararsız (stable/instable) olarak ayrımı diğer bir özelliktir. Algoritmanın birleştirilecek veri yapısı dışında kullandığı ek bellek alanı, algoritmanın kullandığı bellek miktarı diye adlandırılmaktadır. Sabit miktarda ek bellek kullanılması ile *yerinde* (inplace) özelliğine sahip olmaları ve değişken miktarda ek dizin alanına veya imleç (pointer) alanına ihtiyaç duymaları olmak üzere bellek miktarlarına göre temel olarak ikiye ayrılmaktadırlar. Bazı birleştirme algoritmaları giriş dizisinde; en küçük anahtar eleman değerinin başta ve/veya en büyük değerli elemanın sonda bulunmasına ihtiyaç duymaktadırlar. Baştaki ve sondaki bu elemanlara *gözcü* (sentinel) denilmektedir.

Bu tez kapsamında ele alınan tüm birleştirme algoritmaları dizin veri yapısındaki dizileri içsel olarak birleştirmektedirler. Dizi elemanları sadece anahtar elemanlardan oluşmaktadır ve geri kalan bölümlerde aksi belirtilmedikçe dizin elemanı değeri, anahtar elemanı değeri yerine kullanılmıştır. Birleştirme işlemiyle hedeflenen N elemanlı bir L dizininde ardışık olarak bulunan ve eleman değerlerine göre sıralı m elemanlı L[0..m-1] ve n elemanlı L[m..m+n-1], m+n=N, iki diziden oluşan problemi sıralı tek bir dizi haline getirerek çözmektir. Düşey eksen eleman değerleri, yatay eksen dizin pozisyonları olmak üzere ele alınan problemin birleştirme işlemiyle hedeflenen çözümü kavramsal olarak Şekil 2.1'de gösterilmiştir.

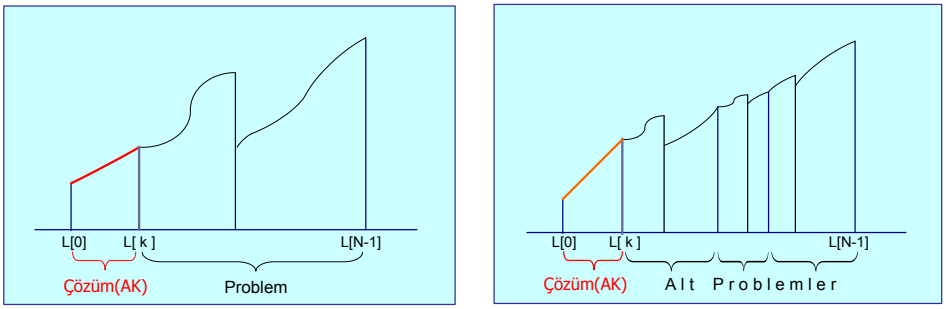
Sıralama algoritmalarında olduğu gibi birleştirme algoritmalarının matematiksel analizi dizi elemanlarındaki taşıma ve anahtar elemanlardaki karşılaştırma işlemi sayıları dikkate alınarak yapılmaktadır. Bu işlemler için alt sınırlar ($m \leq n$ için) sırasıyla $\Omega(n+m)$, $\Omega(m\log(n/m))$ olarak hesaplanmıştır (Knuth, 1973). En iyi (optimum) zaman karmaşıklığı bu karmaşıklıkların doğrusal olduğu zaman -O(N)- olarak alınmaktadır (Geffert et al., 2000).



Şekil 2.1 Ele alınan Problem ve Hedeflenen Çözüm

İlerleyen bölümlerde bazı notasyonlar kullanılmıştır. Bir dizin içinde eleman değerlerine göre sıralı olarak bulunan diziler için *blok*, elemanları ardışık olarak bulunan alt diziler için *kesim (segment)* denmektedir. *D* ve *H* blokları bir *L* dizininin içinde sırasıyla *L[dd..dh]* ve *L[hd..hh]* kesimlerinde bulunmaktadır. Aksi belirtilmedikçe bu bloklar ardışıktır, hd=dh+1, ve *DH* kesimini oluşturmaktadırlar. $|U|$ bir *U* bloğunun/kesiminin eleman sayısını ve U[j] $(1 \leq j \leq |U|)$ bir U bloğundaki/kesimindeki j'inci elemanı göstermektedir.

Birleştirme algoritmaları genel olarak işleyiş bakımında iki farklı yaklaşım üzerine geliştirilmektedirler. Çözüme ulaşmaktaki bu yaklaşımlardan ilki, *sıralı çözüm süreci*, başlangıç problemini küçültürken bir tane olan problem sayısını koruyarak çözüm kümesini büyütmektir. Diğeri ise, *ayrıştırma tabanlı çözüm süreci*, problemi daha küçük alt problemlere bölerek çözmektir. Dizin içinde sıralı/ayrıştırma tabanlı çözüm süreçlerinde işlemin herhangi bir anında çözüm alt kümesi L[0..k] $(0 \leq k \leq N-1)$ ve problem(ler) kalan bölümde bulunmaktadır. Sırasıyla bu durumlar Şekil 2.2.(a) ve Şekil 2.2.(b)'de gösterilmiştir.



(a) Sıralı Çözüm Süreci *(b) Ayrıştırma Tabanlı Çözüm Süreci*

Şekil 2.2 Sıralı-Ayrıştırma Tabanlı Çözüm Sürecinde Anlık Durum

Kararlı ve yerinde bir birleştirme algoritmasını sıralı çözüm süreciyle gerçekleyen optimal zamanlı algoritmalar -Pardo'nun (Pardo, 1977), Huang ve Langston'ın (Huang and Langston, 1992), Symvonis'in (Symvonis, 1995), Geffert et al.'ın (Geffert et al., 2000) ve Chen'in (Chen, 2003) algoritmaları- bulunmaktadır. Fakat bunlar ya algoritmik karmaşıklıklarında büyük sabit çarpanlar olmasından dolayı pratik değildirler ya da çok sayıda kontrol içeren karmaşık yapılarından ve/veya kullandıkları karmaşık tekniklerden dolayı gerçeklenmeleri çok zordur ve etkin değildirler.

Ayrıştırma tabanlı olarak geliştirilen tüm algoritmalar temel olarak iki işlem içeren *ayrıştırma süreci* ile gerçekleştirilmektedir. Ayrıştırma sürecindeki işlemler: problemi oluşturan blokların bir mantığa göre alt blok sınırlarının bulunması **–blok sınırlarının bulunması işlemi-** ve elde edilen alt bloklarda bir takım blok değiştirme **–blok değiştirme işlemi-** tekniklerinin uygulanmasıdır. Bu işlemlerdeki farklılıklar ve diğer bazı performans arttırıcı değişiklikler algoritmalardaki çeşitliliği oluşturur. Basit yapıları sayesinde gerçeklenmesi kolaydır ve etkindir. Blok sınırlarının bulunması işlemi D ve H bloklarının sırasıyla D_1D_2 ve H_1H_2 gibi alt bloklara ayrılmasını içermektedir. Bu işlemdeki mantık D_2 ve H_1

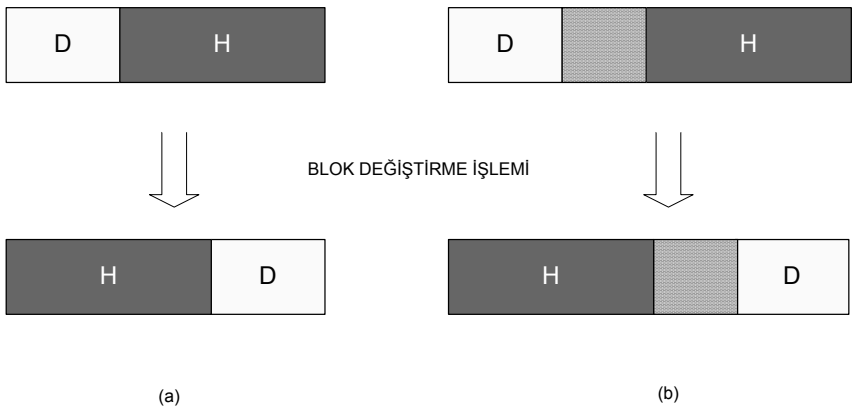
blokların değiştirilmesi sonucunda elde edilen (D_1,H_1) ve (D_2,H_2) alt problemlerin birbirlerinden bağımsız olması, yani alt problemlerdeki elemanların son yerlerinin içerdiği sınırlarda olması gerekmektedir. Blok değiştirme işlemi ile (D,H) problemi, (D_1,H_1) ve (D_2,H_2) bağımsız olan iki alt probleme bölümlenmektedir (partitioning).

2.2 Kullanılan Teknikler

Bu bölümde birleştirme işleminde yaygın olarak kullanılan bazı genel teknikleri açıklanmaktadır. İleriki bölümler için temel teşkil eden bu teknikler detaylarıyla ele alınmıştır.

2.2.1 Blok değiştirme teknikleri

Blok değiştirme işlemi için kullanılan teknikler, ardışık blokların ve/veya bağımsız blokların üzerine uygulanabilir olması durumuna göre sınıflandırılmaktadır. D ve H bloklarının ardışık olması durumundaki değiştirme işlemi Şekil 2.3.(a)'da ve bağımsız olmaları durumundaki işlem ise Şekil 2.3.(b)'de gösterilmektedir.



Şekil 2.3 Blok Değiştirme İşlemi

Bu işlemi gerçekleştiren farklı teknikler ve algoritmalarının analitik sonuçları bölümün devamında anlatılmıştır. Algoritmaların matematiksel analizlerinde atama (ATM), karşılaştırma (KAR), artırma (ART), azaltma (AZL), aritmetik (ARİ) ve dizin elemanı taşınma (TAŞ) işlemleri gözönünde bulundurulmuştur. Dizin elemanlarının taşınması maliyeti u_1 ve diğer tüm işlem maliyetleri u_2 alınarak; u_1 ile u_2 arasındaki oran dizinin eleman boyutu ile tamsayı veri tipinin(C programlama dilinde: int) boyutu arasındaki orana eşit kabul edilecektir (Eşitlik 2.1).

$$u_1 / u_2 = Boyut(D[a])/Boyut(int) \qquad (2.1)$$

2.2.1.1 Karşılıklı Değiş Tokuş tekniği

Karşılıklı değiş tokuş tekniği D ve H bloklarının eleman sayılarının eşit olması durumunda (|D| = |H|) uygulanabilmektedir. Değiştirilecek iki bloğun komşu olması zorunlu değildir. Değiştirme işlemi her D[a]'nın H[a] ile değiştirilmesini içermektedir. Bu tekniğin algoritması kaba kod (pseudocode) olarak Tablo 2.1'de verilmektedir.

```
DegisTokus(D,H)
   1.  for i= 1 to |D|          //|D|=|H|
   2.    begin
   3.      ek_alan =  D[i];
   4.      D[i] = H[i];
   5.      H[i] = ek_alan;
   6.    end.
```

Tablo 2.1 Karşılıklı Değiş Tokuş ile Blok Değiştirme Algoritması

Döngü |D| kez çalışır. TAŞ işlemleri 3,4,5'inci adımlarda toplam 3|D| kez yapılmaktadır. Adım 1'de (|D|)+1 kez KAR ve |D| kez ART olmak üzere 2|D|+1 işlem gerçekleştirilir. Toplam uzunluk N olduğunda algoritmanın toplam maliyeti yaklaşık olarak:

$$T\left[DegisTokus\right] \cong \frac{3N}{2}u_1 + Nu_2,\ |D| = \frac{N}{2} \qquad (2.2)$$

olmaktadır. Dizin elemanlarının tamsayı olduğu durumda tekniğin zaman maliyeti Eşitlik 2.3'te verilmiştir.

$$T\left[DegisTokus\right] \cong \frac{5N}{2}u_1,\ u_1 = u_2 \qquad (2.3)$$

2.2.1.2 Ters Çevirme tekniği

Ters Çevirme tekniği ilgili bloklar/kesimler üzerinde uygulanan bir dizi ters çevirme işlemleriyle gerçekleştirilmektedir. Herhangi bir C blok/kesiminde ters çevirme işlemini Ters(C) ve bu işlem uygulandıktan sonra oluşan C'nin tersini C' ile gösterecek olursak: birbirlerine komşu olan D ve H bloklarının tersleri D' ve H' ile ifade edilir. C_{ilk}=DH kesimi D ve H blokları değiştikten sonra C_{son}=HD biçiminde gösterirsek: ters çevirme işlemleri sırasıyla :

- Ters(D) ---- C_1= D'H
- Ters(H) ---- C_2= D'H'
- Ters(C_2) ---- C_{son}= (D' H')' = HD

uygulanarak blokların değiştirilmesi gerçekleştirilir. Tablo 2.2'de ardışık bloklarda uygulanan bu tekniğin algoritması gösterilmektedir. Bu algoritmada yapılacak değişikliklerle bağımsız bloklar üzerinde bu tekniğin uygulanması sağlanabilinmektedir (Symvonis, 1995).

```
TersCevirme(D,H)
   1. Ters(D);
   2. Ters(H);
   3. Ters(D',H');
```

```
Ters(C)
   1. yorum C bloğu L[k..r] kesitinde bulunuyor.
   2. while k<r do
   3.   begin
   4.     temp = L[k];
   5.     L[k] = L[r];
   6.     L[r] = ek_alan ;
   7.     k = k+1;
   8.     r = r+1;
   9.   endwhile
```

Tablo 2.2 Ters Çevirme ile Blok Değiştirme Algoritması

Bir D bloğunun elemanlarını tersine çevirme işlemi:

$$D[1+i] \leftrightarrow D[|D|-i],\ i = 0,...,|D|/2-1$$

elemanların değiştirilmesi ile gerçekleşir. Her değiştirme işlemi 3 eleman taşıması yaptığından toplam 3|D|/2 tane TAŞ işlemi gerçekleştirilir. Böylece ters çevirme tekniğindeki toplam TAŞ sayısı 3 ters çevirme işleminin toplamına eşit olmaktadır (Eşitlik 2.4).

$$TAŞ[TersCevirme] = \frac{3|D|}{2} + \frac{3|H|}{2} + \frac{3|DH|}{2} = 3|DH| \qquad (2.4)$$

TersCevirme algoritmasında 3 tane *Ters* işlemi yapılmaktadır. *Ters* algoritmasındaki döngü *l* uzunluğundaki bir blok/kesim için $\lfloor l/2 \rfloor$ kere çalışmaktadır. Adım 4,5,6'da gerçekleştirilen TAŞ işlemlerinin toplam sayısı da önceden belirtildiği gibi $3\lfloor(|D|+|H|)/2\rfloor$ olmaktadır. İkinci

adımdaki KAR işlemi sayısı $\lfloor l/2 \rfloor + 1$, adım 7 ve 8'deki ART işlemi $2\lfloor l/2 \rfloor$ kez çalışmaktadır. $\lfloor\ l/2\ \rfloor \cong l/2$ kabul edersek 3 *Ters* çağrımında gerçekleştirilen TAŞ dışındaki temel işlemlerin (KALAN) sayısı toplamı Eşitlik 2.5'de verilmiştir.

$$KALAN[TersCevirme] = 3(|D| + |H| + 1) \cong 3(|D| + |H|) \qquad (2.5)$$

Toplam uzunluğu N olarak alırsak, Eşitlik 2.4 ve 2.5 kullanılarak algoritmanın toplam maliyetini Eşitlik 2.6'daki gibi elde ederiz.

$$T[TersCevirme] \cong 3N(u_1 + u_2),\ |DH| = N \qquad (2.6)$$

2.2.1.3 Ters Permütasyon tekniği

Dudzinski ve Dydek tarafından geliştirilmiş, verilen iki komşu bloğun değiştirilmesinde sadece |DH| + OBEB(|D|,|H|) –OBEB: ortak bölenlerin en büyüğü- eleman taşıması gerçekleştiren diğerlerinden daha karmaşık bir tekniktir (Dudzinski and Dydek, 1981). Teknik temel olarak dizin elemanlarının blok değiştirme işlemi sonrasında başlangıç yerlerinin indisleri tarafından ifade edilebilen bir permütasyon fonksiyonunun tersine dayanmaktadır. Bu σ permütasyon foksiyonu :

$$\sigma = \begin{pmatrix} 1 & 2 & \ldots\ldots\ldots & |D| & |D|+1 & |D|+2 & \ldots\ldots & |D|+|H| \\ |H|+1 & |H|+2 & \ldots\ldots\ldots & |D|+|H| & 1 & 2 & \ldots\ldots & |H| \end{pmatrix}$$

ile ifade edilir. σ'(x) ters permutasyonu ise Eşitlik 2.7'de gösterilmiştir.

$$\sigma'(x) = (x + |D|)\ mod(|DH|) \qquad (2.7)$$

Tablo 2.3'de Ters Permütasyon tekniğinin algoritmasının kaba kodu verilmektedir.

```
TersPermutasyon(D,H)
   1.    P = GCD(|D|,|H|);
   2.    yorum P: Ortak Bölenlerin En Büyüğü;
   3.    for t=1  to  P
   4.      begin
   5.        hedef = t  ;
   6.        kaynak = |D|+t ;
   7.        ek_alan = DH[hedef];
   8.        while kaynak != t  do
   9.          begin
   10.             DH[hedef] = DH[kaynak];
   11.             destination = kaynak ;
   12.             kaynak = σ'(kaynak);
   13.             yorum  σ': x tane döngüye sahip ;
   14.             yorum  permütasyon fonksiyonu ;
   15.         end.
   16.       DH[hedef]  = ek_alan ;
   17.     end.
```

Tablo 2.3 Ters Permütasyon ile Blok Değiştirme Algoritması

σ'(x) fonksiyonu, P = OBEB(|D|,|H|) olmak üzere DH kesiminde her birinin uzunluğu P olan |DH|/P adet döngü oluşturmaktadır. Her döngü içinde x indisine gelecek eleman, σ'(x) ile hesaplanarak son yerine tek eleman taşıması ile ulaşmaktadır. Her döngünün başında ilk yerdeki eleman ek bir alana alınarak döngü sona erdiğinde en son yere taşınır. *P* tane döngü için her bir döngüde $|DH|/P+1$ taşıma gerçekleşeceğinden toplam taşıma sayısı :

$$TAŞ[TersPermutasyon] = P\left(\frac{|DH|}{P}+1\right) = |DH|+P \qquad (2.8)$$

eşit olur. Değiştirme işleminde blokların herhangi birisinde eleman sıralarının korunması gerekmiyor ise toplam taşıma sayısı azaltılabilir. D ve H bloklarından: büyük olan tarafın korunması gerekiyorsa toplam taşıma sayısı $|DH|+1$, küçük olanın korunması gerekiyorsa $2\min(|D|,|H|)+1$'e düşürülebilmektedir (J.Chen, 2003).

TersPermutasyon algoritmasında dıştaki döngü *P* kere çalışırken içteki döngü $|DH|-P$ kere çalışmaktadır. σ'(x) (Bkz. Eşitlik 2.7) fonksiyonunda toplam 2 tane ARİ bulunmaktadır. *P* kere çalışan dıştaki döngüde, adım 3'de 1 ART ve 1 KAR; adım 5 ve 7'de 1'er ATM; adım 6'da 1 ATM ve 1 ARİ olmak üzere toplam 6P işlem gerçekleşmektedir. $|DH|-P$ kere çalışan içteki döngüde, adım 8'de 1 KAR; adım 11'de 1 ATM; ve adım 12'deki 1 ATM ve 2 ARİ işlem toplam 5($|DH|-P$) işlem yapılmaktadır. Böylece OBEB alt yordamında toplam X tane işlem yapılıyor varsayarsak, taşıma dışında kalan diğer temel işlemlerin toplamı:

$$KALAN[TersPermutasyon] = X + 5|DH| + P \qquad (2.9)$$

olur. |DH| uzunluğunu N alırsak, Eşitlik 2.8 ve Eşitlik 2.9'dan algoritmanın maliyetini Eşitlik 2.10'daki gibi elde ederiz.

$$T[TersPermutasyon] \cong Nu_1 + (5N + P + X)u_2,\ |DH| = N \qquad (2.10)$$

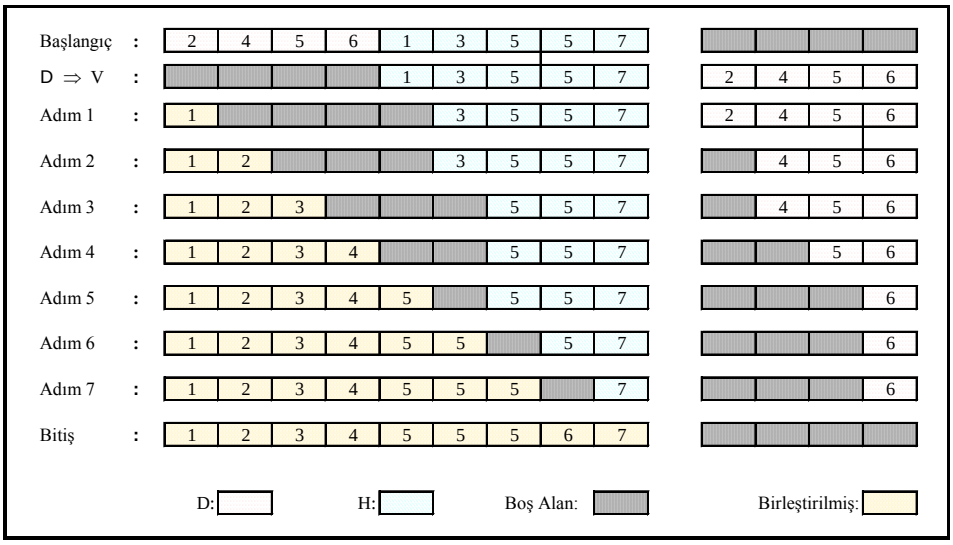
2.2.2 Basit Birleştirme Algoritması

Bu bölümde, m ve n elemanlı iki bloğu O(min(m,n)) ek bellek kullanarak O(m+n) karşılaştırma ve O(m+n) taşıma işlemi yaparak birleştiren basit bir algoritma (Basit Birleştirme Algoritması, BBA) detaylı olarak anlatılmıştır (Symvonis, 1995).

Algoritmada ilk olarak L[dd...dh] kesiminde bulunan m elemanlı D bloğu ve L[dh+1...hh] kesimindeki n elemanlı H bloğundan, eleman sayısı az olan seçilerek, tüm elemanları aynı boyutu -min(m,n)- sahip olan V ek bellek alanına taşınmaktadır. m ≤ n için V [0..m-1] alanı D bloğunun elemanlarından oluşmaktadır. İşlem sırasında a tane D bloğuna ait elemanla b tane H bloğuna ait elemanın birleştirildiğini düşünürsek:

- Birleştiren elemanlar L[dd...dd+a+b-1] kesiminde bulunur.
- D bloğunda kalan elemanlar V[a...m-1] kesiminde bulunur.
- H bloğunda kalan elemanlar L[dh+b...hh] kesiminde bulunur.

Bundan sonraki adımda, L[dh+b] pozisyonundaki H'nin elemanı ile V[a] pozisyonundaki D'nin elemanı karşılaştırılarak: küçük ise D'nin, değilse H'nin elemanı L[dd+a+b] pozisyonuna taşınmaktadır. İşlem V alanındaki elemanlar bitene kadar devam eder. Önce H bloğundaki kalan elemanların bitmemesi için en sonda gözcü elemana ihtiyaç vardır. Şekil 2.4'de algoritmanın işleyişi bir örnek üzerinden gösterilmiştir.

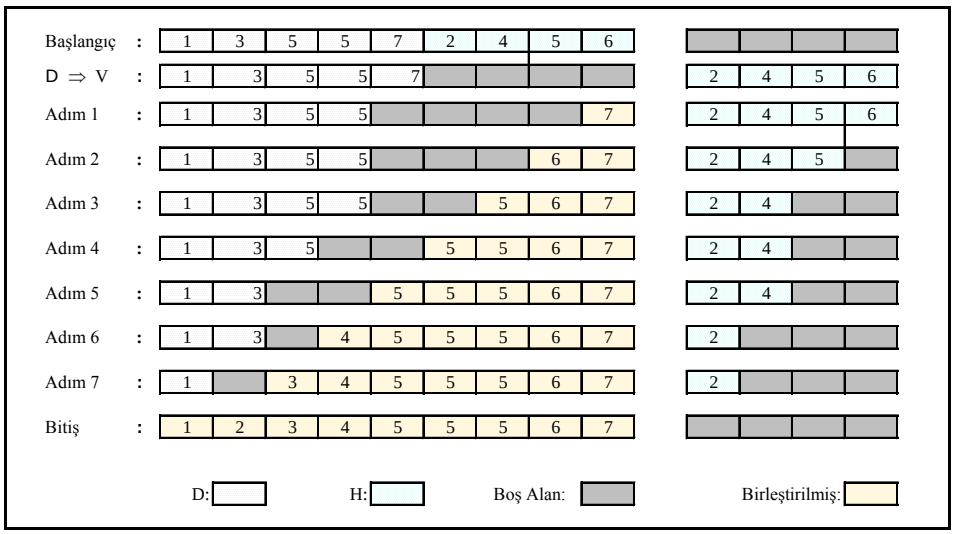


Şekil 2.4 Basit Birleştirme Algoritması Örnek-1

n<m olması durumunda ise H bloğunun elemanları V[0...n-1] alanına taşınır. İşlem sırasında a tane D bloğuna ait elemanla b tane H bloğuna ait elemanın birleştirildiğini düşünürsek:

- Birleştiren elemanlar L[hh-a-b+1...hh] kesiminde bulunur.
- H bloğunda kalan elemanlar V[0...m-b-1] kesiminde bulunur.
- D bloğunda kalan elemanlar L[dd...dh-a] kesiminde bulunur.

Bundan sonraki adımda, L[dh-a] pozisyonundaki D'nin elemanı ile V[m-b-1] pozisyonundaki H'nin elemanı karşılaştırılmaktadır: büyük ise D'nin, değilse H'nin elemanı L[hh-a-b] pozisyonuna taşınır. İşlem V alanındaki elemanlar bitene kadar devam eder. Önce D bloğundaki kalan elemanların bitmemesi için en başta bir gözcü elemana ihtiyaç vardır. Şekil 2.5'de algoritmanın işleyişi bir örnek üzerinde gösterilmiştir.



Şekil 2.5 Basit Birleştirme Algoritması Örnek-2

2.2.3 İkili Arama ve eleman ekleme teknikleri

Bu bölüm, ikili arama tekniğinin değişik uygulamalarını içerir. Arama kriterinde hedeflenen, aranan elemanın sıralı bir seri içinde ekleneceği pozisyonun bulunmasıdır. Arama kriterindeki değişimin işlemin sonucunda etkisi bulunmaktadır. N elemanlı sıralı bir L dizinine x değerine sahip bir elemanın ekleneceğini düşünelim. Değişen kritere göre ikili aramayla L dizininde x'in ekleneceği i pozisyonunu bulan genel algoritmanın sözde kodu Tablo 2.4'de verilmiştir.

```
İKİLİ_ARAMA_GENEL(x,L)
   1.   integer i ;
   2.   integer küçük = 0;
   3.   integer büyük = N-1;
   4.   while küçük ≤ büyük do
   5.     begin
   6.       i=⌈(küçük+büyük)⌉/2;
   7.       if <ARAMA KRİTERİ> then
   8.          küçük = i+1;
   9.       else büyük = i-1;
   10.    endwhile
   11.    i= küçük;
   12.   yorum i: x'in pozisyonunu tutuyor.
```

Tablo 2.4 Genel İkili Arama Algoritması

Hedefimiz ilk olarak bu elemanın ekleneceği pozisyondan önceki tüm elemanların değerlerinin x'ten küçük olması olsun. Bunu gerçekleştiren algoritmaya da İA1 diyelim. Böylece İA1 ile sağlanması gereken durum :

$L[1...i-1] < x$ *ve* $L[i...N] \geq x$

olmaktadır. *ARAMA KRİTERİ* olarak L[i] < x alınırsa x'in pozisyonu ikili

aramayla bulunmaktadır. Eğer hedefimiz eklenecek elemanın pozisyonundan sonraki tüm elemanların x'ten büyük ve öncekilerinde x'e küçük veya eşit olması olursa , ve bunu gerçekleştiren algoritmayı İA2 diye adlandırırsak sağlanması gereken durum :

$$L[0...i-1] \leq x \text{ ve } L[i...N-1] > x$$

olmaktadır. *ARAMA KRİTERİ* olarak L[i] ≤ x alınırsa, x'in pozisyonu ikili aramayla bulunmaktadır.

Tanımlanan problemi, N elemanlı L_1 [0...N-1] ve L_2 [0...N-1] sıralı dizileri için bulunması gereken durum:

$$L_2[k] < L_1[N-1-k] \quad , k= 0 \ldots\ldots i-1$$

$$L_2[j] \geq L_1[N-1-j] \quad , j= i \ldots\ldots N-1$$

olarak düşünecek olursak: genel algoritmada tanımlanan x yerine L_1 ve L yerine L_2 alındığında , *'ARAMA KRİTERİ'* olarak ta L_2[i] < L_1[N-1-i] seçildiğinde istenilen durumumu gerçekleyen İA3 algoritması elde edilmektedir.

İkili aramaya dayanan İA1,İA2,İA3 algoritmalarında pozisyonun bulunması işlemi O(logN) eleman karşılaştırma işlemi ile gerçekleştirilmektedir. İA1 ve İA2 algoritmaları sonundaki amaç sadece x değerine sahip elemanı L dizinine eklemek olsaydı m tane elemanı bu dizine eklemek için yapılan toplam karşılaştırma O(mlogN) olurdu. Hwang ve Lin tarafından geliştirilen değiştirilmiş bir ikili arama tekniğiyle bu karşılaştırma karmaşıklığı m≤N için O(mlog(N/m)) olarak gerçeklenmiştir (Hwang and Lin, 1972). *"İkili-gibi Arama"* diye adlandırılan bu teknik N elemanlı sıralı diziyi N/m elemanlı m tane diziye bölüp, ikili arama yapılacak alt diziyi bulma temeline dayanmaktadır (Symvonis, 1995).

2.3 İlgili Algoritmalar

Bu bölümde literatürde bulunan ayrıştırma tabanlı kararlı birleştirme algoritmaları anlatılmıştır.

2.3.1 Dudzinsky ve Dydek'in algoritması

Bu algoritma m ve n uzunluktaki (m ≤ n için) sıralı iki dizini, O(log m) ek bellek kullanarak O((m+n)log m) eleman taşıması ve O(m(log(n/m+1))) karşılaştırması işlemi kullanarak birleştirir (Dudzinsky and Dydek, 1981). Özyineleme (recursion) özelliğine sahiptir. Sonraki bölümlerin anlaşılmasını kolaylaştırmak için bu algoritma kısaca AT1 olarak adlandırılmıştır. Tablo 2.5'te AT1'in kaba kodu verilmiştir.

AT1'de, ayrıştırma tabanlı algoritmalarda temel olan iki işlem: değiştirilecek blok sınırlarının bulunması ve blok değiştirme için sırasıyla ikili arama ve ters permütasyon teknikleri kullanılmaktadır. Değiştirilecek blok sınırlarının bulunması işlemi herhangi sıralı D ve H bloklarının; D_1, D_2, H_1, H_2 alt bloklarına ve x elemanına aşağıdaki sürece göre ayrılması ile gerçekleştirilir:

(a) Eğer $|D| \le |H|$ ise

- *D*'nin orta elemanı x iken $D = (D_1 \text{ x } D_2)$ biçiminde ayrılır: $|D_1| = \lfloor |D|/2 \rfloor$ ve $|D_2| = \lfloor (|D|-1)/2 \rfloor$
- *H* bloğu İA1 ikili arama algoritmasıyla $H = (H_1\ H_2)$, H_1'in tüm eleman değerleri x'ten küçük olacak ve H_2'nin tüm eleman değerleri x'ten küçük olmayacak, biçimde ayrılır.

(b) Eğer |*D*| > |*H*| ise

- *H*'nin orta elemanı x iken $H = (H_1 \text{ x } H_2)$ biçiminde ayrılır: $|H_1| = \lfloor |H|/2 \rfloor$ ve $|H_2| = \lfloor (|H|-1)/2 \rfloor$

- *D* bloğu İA2 ikili arama algoritmasıyla $D = (D_1\ D_2)$, D_1'in tüm eleman değerleri x'ten büyük olmayacak ve D_2'nin tüm eleman değerleri x'ten büyük olacak, biçimde ayrılır.

```
Algoritma  AT1(D,H)
1 yorum  D=L[dd:hd-1] ve H=L[hd:hh]
2 begin
3  if |D| ≠ 0 and |H| ≠ 0  then
4   if |D| ≤ |H| then
5     s = ⌈((dd+hd-1) / 2)⌉;
6       yorum x=L[s], D1=L[dd..s-1] ve D2 = L[s+1..dh];
7     i= İA1(L[s],H);
8       yorum  H1=L[hd..i-1] ve H2= L[i..hh];
9     TersPermutasyon(xD2,H1);
10  else
11     s = ⌈((hd+hh) / 2)⌉ ;
12      yorum x=L[s], H1=L[hd..s-1] ve H2 = L[s+1..hh];
13    i = İA2(L[s],D);
14      yorum  D1=L[dd..i-1] ve D2= L[i..dh];
15    TersPermutasyon(D2,H1x);
16  endelse;
17    yorum   DH = (D1H1) x (D2H2) olarak değiştiriliyor.
18  AT1(D1,H1); AT1(D2,H2);
19  endif;
20 end;
```

Tablo 2.5 AT1 Algoritması

Değiştirilecek blokların bulunması işleminden sonra ters permütasyon tekniğinin uygulanması işlemi:

(a) Eğer $|D| \le |H|$ ise

- xD_2 ile H_1 blokları değiştirilir.

(b) Eğer $|D| > |H|$ ise

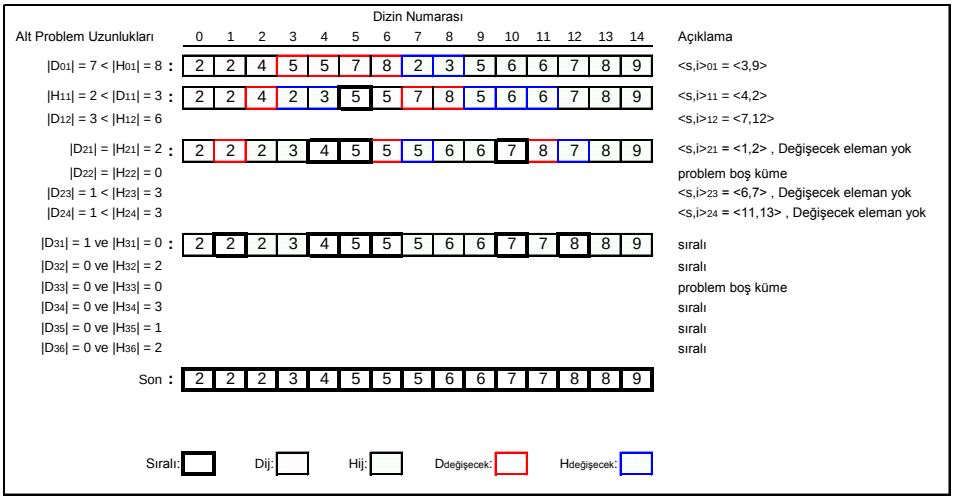
- D_2 ile H_1x blokları değiştirilir.

olarak gerçekleştirilmektedir. Tüm bu işlemler sonucunda DH kesimi :

$$DH = (D_1H_1)x(D_2H_2)\text{ , maks}(D_1H_1) \le x \text{ ve min}((D_2H_2) \ge x\ )$$

biçiminde iki bağımsız alt probleme ayrılmaktadır. Her problem parçacığında –önce baştaki sonra sondaki alt problem olmak üzere- algoritma özyinelemeli olarak aynı şekilde çalışmaktadır.

Şekil 2.6'da AT1 algoritmasının işleyişi bir örnek problem üzerinde gösterilmiştir. Ayrıştırma işlemleri, bölümlenme sonrası ele alınan yeni alt problemler yerine –alt problemlerin çözüm sırası yerine- özyinelemenin her adımındaki alt problemlerde gösterilmiştir.



Şekil 2.6 AT1 Algoritması Örnek

D_{dj} ve H_{dj} özyinelemenin d'inci adımındaki j'inci alt problemi oluşturan blokları, $<s,i>_{dj}$ ise bu alt problemdeki küçük bloğun orta elemanının pozisyonunu (s) ve ikili arama sonucu dönen pozisyon değerini (i) belirtmektedir (Bkz. Tablo 2.5). Başlangıçta, sıralı D_{01} ve H_{01} dizileri, L[0..14] dizininde sırasıyla L[0..6] ve L[7..14] kesimlerinde bulunmaktadır. $|D_{01}| = 7 < |H_{01}| = 8$ olduğu için s değeri D_{01} bloğunun orta eleman pozisyonuna -3- eşittir ve değiştirilecek kesim L[3..6]'da

bulunmaktadır. L[3] = 5 iken H_{01} bloğunda gerçekleştirilen ikili arama sonucu dönen değer i = 9 olurken değiştirilecek kesim L[7..8] olmaktadır. Bu iki kesimin (alt blok) ters permütasyon tekniği kullanılarak değiştirilmesi ile başlangıç problemi, biri L[0..4] kesiminde -D_{11}: L[0..2]: ve H_{11}: L[3..4]- bulunan diğeri ise L[6..14] kesiminde -D_{11}: L[6..8]: ve H_{11}: L[9..14]- bulunan iki alt probleme ayrılmıştır. Bu alt problemler ve bunlardan üretilen alt problemler üzerinde algoritmanın işleyişi Şekil 2.6'da gösterilmiştir.

AT1 algoritmasında m ve n uzunluklarının, m = n = N/2 olması durumunda gerçekleşecek eleman taşıma ve karşılaştırma işlemlerinin sayısı sırasıyla Eşitlik 2.11'de verilmiştir.

$$TAŞ[AT1] = \mathrm{O}(N \log N) \text{ ve } KAR[AT1] = \mathrm{O}(N) \quad (2.11)$$

2.3.2 Mehmet Emin Dalkılıç'ın algoritması

Bu algoritma m ve n uzunluktaki (m ≤ n) sıralı iki diziyi, en kötü durumda O(n) ek bellek kullanarak O(m+n)+O(mlog m) eleman taşıma ve O(m+n) karşılaştırma karmaşıklığıyla birleştirir (Dalkılıç, Unpublished). Karşılaştırma ve taşıma zaman karmaşıklıklarının ispatı Ek 1'de verilmiştir. Özyinelemeli olarak çalışan gerçeklenmesi kolay bir algoritmadır. Sonraki bölümlerin anlaşılmasını kolaylaştırmak için bu algoritma AT2 olarak adlandırılmıştır. Tablo 2.6'da algoritmanın kaba kodu verilmiştir.

Tablo 2.6'dan görüleceği gibi sadece 2 temel işlem'den oluşmaktadır. Adım 4'te gerçekleştirilen İA3 ikili arama algoritmasıyla değişecek bloklar bulunmaktadır. İA3 ile bulunan i değeri D ve H bloklarında orta noktaya eşit uzaklıkta olan ve L[dh-i] ≤ L[hd+i] durumunu sağlayan ilk elemanların bağıl indisleridir.

```
Algoritma  AT2(D,H)

   1.  yorum  D=L[dd..hd-1] ve H=L[hd..hh]
   2.  begin
   3.   if |D| ≠ 0 and |H| ≠ 0  then
   4.      i= İA3(D,H);
   5.        yorum  D1=L[dd..dh-i] ve D2= L[dh-i+1..dh]
   6.               H1=L[hd..hd+i-1] ve H2= L[hd+i..hh];
   7.      DeğişTokuş(D2,H1);
   8.     yorum    DH = (D1H1)(D2H2) oluyor
   9.   AT2(D1,H1); AT2(D2,H2);
   10.  endif;
   11. end;
```

Tablo 2.6 AT2 Algoritması

Bağıl indis değerine (i) göre D ve H kesiminde elde edilen $D_1D_2H_1H_2$ yeni blok parçacıkları L[dd..hh] dizininde:

- D_1=L[dd..dh-i] ve D_2= L[dh-i+1..dh]
- H_1=L[hd..hd+i-1] ve H_2= L[hd+i..hh]

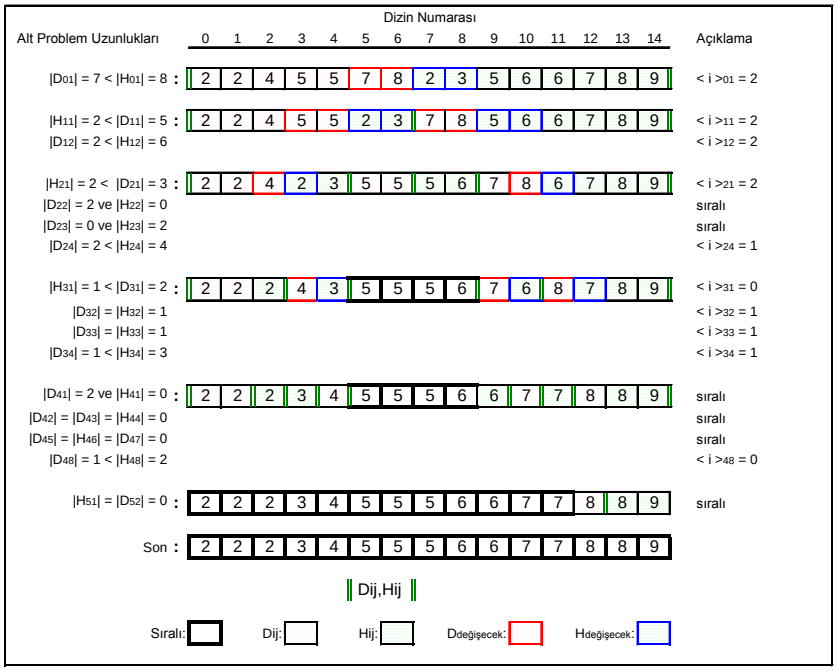
kesimlerinde bulunmaktadırlar. D_2 ve H_1 eleman sayıları birbirine eşit olup i'ye eşittir. Gerçekleştirilen ikinci işlem adım 7'deki bu iki bloğun karşılıklı değiş tokuş tekniği ile değiştirilmesidir. H_1'deki elemanların en büyük değere sahip olanı, (H[hd+i-1]) D'deki elemanların en küçük değere sahip olanından (D[dh-i+1]) her zaman büyük olacağı için kararlılık özelliği sağlanmış olmaktadır. Blok değiştirme işlemi sonucunda L[dd..hh] kesimi, D_1=L[dd..dh-i], H_1=L[dh-i+1..dh], D_2=L[hd..hd+i-1] ve H_2=L[hd+i..hh] bloklarından oluşmaktadır. Oluşan bu (D_1H_1) ve (D_2H_2) yeni kesimlerinde:

$$\text{maks}\,(D_1H_1) \leq \min(D_2H_2)$$

durumu sağlanarak iki yeni ve bağımsız alt problem elde edilmektedir.

Her alt problemde AT2 algoritması yeni problem bloklarından birisinin elemanı kalmayıncaya kadar devam eder.

Şekil 2.7'de AT2 algoritmasının işleyişi bir örnek problem üzerinde gösterilmiştir. Ayrıştırma işlemleri, bölümlenme sonrası ele alınan yeni alt problemler yerine –alt problemlerin çözüm sırası yerine- özyinelemenin her adımındaki alt problemlerinde gösterilmiştir.



Şekil 2.7 AT2 Algoritması Örnek

D_{dj} ve H_{dj} özyinelemenin d'inci adımındaki j'inci alt problemi oluşturan blokları, $< i >_{dj}$ ise bu alt problemde ikili arama sonucu dönen değiştirilecek eleman sayısını (i) belirtmektedir (Bkz. Tablo 2.6). Başlangıçta sıralı D_{01} ve H_{01} dizileri, L[0..14] dizininde sırasıyla L[0..6] ve L[7..14] kesimlerinde bulunmaktadır. D_{01} ve H_{01} bloklarında gerçekleştirilen ikili arama sonucunda değiştirilecek eleman sayısı iki

olarak (i = 2, L[4] = 5 ≤ L[9] = 5) bulunmaktadır. Bunun sonucunda, değiştirilecek kesimler D_{01} bloğunda L[5..6] ve H_{01} bloğunda L[7..8] olmaktadır Bu iki kesimin (alt blok) değiş tokuş tekniği kullanılarak değiştirilmesi ile başlangıç problemi, biri L[0..6] kesiminde -D_{11}: L[0..4]: ve H_{11}: L[5..6]- bulunan diğeri ise L[7..14] kesiminde -D_{11}: L[7..8]: ve H_{11}: L[9..14]- bulunan iki alt probleme ayrılmıştır. Bu alt problemler ve bunlardan üretilen yeni alt problemler üzerinde algoritmanın işleyişi Şekil 2.7'de gösterilmiştir.

AT2 algoritmasında m ve n uzunluklarının, m = n = N/2 olması durumunda gerçekleşecek eleman taşıma ve karşılaştırma işlemlerinin karmaşıklığı Eşitlik 2.12'de verilmiştir.

$$TAŞ[AT2] = O(N \log(N)) \text{ ve } KAR[AT2] = O(N) \qquad (2.12)$$

2.3.3 Dvorak ve Durian'ın algoritmaları

Dvorak ve Durian, ilk olarak AT1 algoritmasında yaptıkları bir takım değişikliklerle çalışma performansını arttırmışlardır. Bu yapılan değişiklikler blok sınırlarının bulunması işleminde, blok değiştirme tekniğinde ve sonda daha hızlı bir birleştirme algoritması kullanılması olmak üzere üç başlık altında toplanmıştır (Dvorak and Durian, 1988). Kalan bölümlerde bu algoritma ATD1 diye adlandırılmıştır.

ATD1'de ilk olan değişiklik D ve H bloklarının sıralı olduğu durumda, D[son] ≤ H[ilk], ayrıştırma işlemi yapılmadan algoritmanın sonlanmasıdır. Blok sınırlarının bulunması işleminde ise orta eleman x'e göre değiştirilecek alt bloğun ikili aramayla bulunması işlemi sadece böyle bir alt bloğun olması durumunda gerçekleştirilerek gereksiz adımlar ortadan kaldırılmıştır. Alt blokların var olup olmaması kontrolü süreci D ve H bloklarının farklı uzunluk eşitliklerine göre Tablo 2.7'de gösterilmiştir.

$\|D\| \le \|H\|$	• Eğer $x \le H[1] \Rightarrow \langle H_1 = 0 \mid H_2 = H \rangle$ • Eğer $x > H[\|H\|] \Rightarrow \langle H_1 = H \mid H_2 = 0 \rangle$ • Aksi halde $H[1] < x \le H[\|H\|] \Rightarrow \exists t \in [1, \|H\|]$ için $H[t] < x \le H[t+1]$
$\|H\| < \|D\|$	• Eğer $x < D[1] \Rightarrow \langle D_1 = 0 \mid D_2 = D \rangle$ • Eğer $x \ge D[\|D\|] \Rightarrow \langle D_1 = D \mid D_2 = 0 \rangle$ • Aksi halde $D[1] < x \le D[\|D\|] \Rightarrow \exists t \in [1, \|D\|]$ için $D[t] \le x < D[t+1]$

Tablo 2.7 ATD1 Alt-Blok Kontrol Süreci

İkinci değişiklik ise blok değiştirme tekniğinde yapılmıştır. Gerek matematiksel analizlerinden gerekse gerçek çalışma sürelerinden ortaya çıkan sonuç TAŞ maliyetinin diğer temel işlem (KAR, ART, AZM vb.) maliyetlerine yakın olduğu durumlarda ters çevirme yönteminin ters permütasyon yönteminden hızlı çalışmasıdır (Dvorak and Durian, 1988). Eşitlik 2.6, 2.10'u ve maliyetleri $u_1 \cong u_2$ için u aldığımızda çıkan matematiksel hızları Eşitlik 2.13'de gösterilmiştir.

$$T[TersPermütasyon] = (6N+P+X)\,u \ge 6Nu = T[TersÇevirme] \quad (2.13)$$

Son değişiklik ise bloklardan birinin uzunluğunun belli bir değerin altına düştüğü zaman daha hızlı çalışan bir birleştirme algoritması kullanılmasıdır. Bu değişikliğin daha hızlı bir algoritma çalıştırmasıyla birlikte diğer bir yararı da bölümleme (partitioning) işleminin belli bir değerde kesilmesidir. *Quicksort* sıralama algoritması için bu belli değerde bölümleme kesildiğinde performansı arttığı kanıtlanmıştır (Dvorak and Durian, 1988).

ATD1'de kesme algoritması olarak gözcü elemanlara ihtiyaç duyan basit birleştirme algoritması kullanılmaktadır. Bu algoritma uzunluğu

küçük olan blok kadar ek bellek maliyeti getirmektedir. Kesme değeri olarak, KESME diyelim, sabit bir değer kullanıldığında ve küçük bloğun eleman sayısı bu değerin altına düştüğünde çalıştığı için sabit miktarda O(KESME) ek belleğe ihtiyaç duyar. Algoritmanın eleman taşıma ve karşılaştırma karmaşıklıkları AT1 algoritmasıyla aynıdır (Dvorak and Durian, 1988).

Dvorak ve Durian'ın yaptığı ikinci çalışma geliştirmiş oldukları özyinelemeli olarak çalışan ATD1 algoritmasını sıralı (sequential) olarak çalışır hale getirmeleridir (Dvorak and Durian, 1988). Bu değişiklikle özyinelemeden dolayı ortaya çıkan bellek kullanım miktarından kurtulmuş olunmaktadır. Bellek karmaşıklığı O(KESME) = O(1) olarak sabit bir değere eşitlenerek optimum değere ulaşmıştır. Hem kararlı hem de yerinde özelliğe sahip bu algoritma ATY1 olarak adlandırılmıştır.

ATY1 algoritmasında problemin iki alt probleme bölünmesi süreci ATD1'in aynısıdır. Farklı olarak her iki probleminde sırayla çözülmesi yerine sadece baştaki alt problem çözülmektedir. Derinlik öncelikli (Depth-First) bu yaklaşım bir döngü içinde derinlikteki alt problem bitinceye yani sıralı dizi elde edilinceye kadar devam eder. Bu derinlik tabanlı çözüm süreci:

- DH sırasızken tekrarlanan işlemler
    - ATD1 ayrıştırma süreci
    - $D = D_1$ ve $H = H_1$

olarak gerçekleşmektedir. Bu süreç sonucunda kalan bölüm problem parçacıklarından oluşmaktadır. Kalan L[i..j] kesiminde:

- D bloğu
    - Başlangıç ve bitiş indisleri dd=dh= i olarak ilklenir.
    - $L[dh] \leq L[dh+1]$ olduğu sürece dh=dh+1.

- L[dd..dh] kesimi olarak seçilir.

- H bloğu
  - Başlangıç ve bitiş indisleri hd=hh=dh+1 olarak ilklenir.
  - L[dh] ≥ L[hh] olduğu sürece hh=hh+1.
  - x = hh ve hh= hh - 1, L[hd..hh] kesimi seçilir

ilk alt problem, L[dd..dh] kesiminde D bloğu ve L[hd..hh] kesiminde H bloğu olmak üzere seri karşılaştırmalarla bulunur. x değeri önceki bölünme sırasında yeri sabit kalan orta elemanın indisidir. Bu yeni problemde derinlik tabanlı çözüm sürecinden geçmektedir. Derinlik tabanlı çözüm süreci ve kalan bölümde ilk alt problemin bulunması süreci tüm dizin sıralanıncaya kadar birbiri ardına uygulanarak sürmektedir.

ATD1 algoritmasının yerinde haline getirilmesiyle geliştirilen ATY1 algoritmasında ortaya çıkan tek ek maliyet blok sınırlarının bulunması için gerçekleşen karşılaştırma işlemleridir. Derinliğin her seviyesinde alt problemlerden yarısının blok sınırları bulunacağı için, bir seviyedeki ek karşılaştırma işlemleri $m/2+n$'ni geçemez. Tüm seviyelerdeki ($\log m$) toplam sayı ise $(m/2+n)\log m$ geçemez

Böylece ATY1 algoritmasında gerçekleşen karşılaştırma işlemleri :Eşitlik 2.14'deki gibi olmaktadır.

$$KAR[ATY1] = O(m\log(n/m+1)+(m/2+n)\log m) \qquad (2.14)$$

$m \cong n \cong N/2$ olduğu durumda ise taşıma işlemleri – AT1 ile aynı Eşitlik 2.12- ve karşılaştırma işlem sayıları Eşitlik 2.15'te gösterilmiştir.

$$T[TAŞ] = T[KAR] = O(N\log N) \qquad (2.15)$$

3 GELİŞTİRİLEN BİRLEŞTİRME ALGORİTMALARI

Bu bölümde geliştirilen ayrıştırma tabanlı birleştirme algoritmaları iki başlık altında toplanmıştır. İlk olarak seri ortamlar için uygun algoritmaların işleme süreçleri ayrıntısıyla ele alınmıştır. Daha sonra ise geliştirilen yerinde özelliğe sahip ayrıştırma tabanlı birleştirme algoritmasının paralel ortamlarda verimli bir şekilde çalışması için bir yöntem sunulmuştur.

3.1 Seri Ayrıştırma Tabanlı Algoritmalar

Temel olarak algoritmalar bir önceki bölümde açıklanan Dalkılıç'ın (Unpublished) özyinelemeli kararlı algoritmasına dayanmaktadır. İlk algoritma, ATD2, mevcut çalışma üzerinde yapılan değişikliklerle zaman performansının arttırılmasıyla geliştirilmiştir. Diğer algoritma -ATY2- ise özyinelemeli kararlı ATD2 algoritmasının sıralı hale getirilmiş yerinde sürümüdür.

AT2 algoritmasında yapılan değişiklikler: ayrıştırma sürecinin alt-blok sınırlarının bulunması işleminin iyileştirilmesi ve sonda daha hızlı bir birleştirme algoritması kullanılmasıdır. ATD2'de gerçekleştirilen ayrıştırma işleminde değiştirilecek alt blokların (D_2 ve H_1) ikili aramayla bulunması işlemi sadece böyle alt blokların olması durumunda gerçekleştirilerek gereksiz adımlar ortadan kaldırılmıştır. Alt blokların var olup olmaması kontrol süreci D ve H bloklarının durumlarına göre Tablo 3.1'de gösterilmiştir.

Son değişiklik ise bloklardan birinin uzunluğunun belli bir değerin altına düştüğü zaman daha hızlı çalışan bir birleştirme algoritması kullanılmasıdır. Bu değişikliğin yararları Bölüm 2.3.3'te anlatılmıştır. Kullanılan kesme algoritması ATD1 algoritmasında da kullanılan basit birleştirme algoritmasıdır. Bu algoritma uzunluğu küçük olan blok kadar

ek bellek maliyeti getirir. Kesme değeri olarak, KESME diyelim, sabit bir değer kullanıldığından ve küçük blok bunun altına düştüğünde çalıştığı için O(KESME) ek bellek miktarı da eklenerek, ATD2'in kullandığı toplam ek bellek miktarı Eşitlik 3.1'de verilmiştir

$$ATD2[Ek\_Bellek] = O(n + kesme\_birleş) = O(n) \qquad (3.1)$$

$\|D\| \le \|H\|$	• Eğer $D[1] > H[\|D\|] \Rightarrow \langle D_2 = D \mid H_1 = H[1..\|D\|] \rangle$ • Eğer $D[\|D\|] < H[1] \Rightarrow \langle D_2 = 0 \mid H_1 = 0 \rangle$ • Aksi halde $\exists t \in [1, \|D\| - 1]$ için $D[\|D\| - t + 1] > H[t]$ ve $D[\|D\| - t] \le H[t+1]$
$\|H\| < \|D\|$	• Eğer $D[\|D\| - \|H\| + 1] > H[\|H\|] \Rightarrow \langle D_2 = D[\|D\| - \|H\| + 1..\|D\|] \mid H_1 = H \rangle$ • Eğer $D[\|D\|] < H[1] \Rightarrow \langle D_2 = 0 \mid H_1 = 0 \rangle$ • Aksi halde $\exists t \in [1, \|H\| - 1]$ için $D[\|D\| - t + 1] > H[t]$ ve $D[\|D\| - t] \le H[t+1]$

Tablo 3.1 ATD2 Alt-Blok Kontrol Süreci

ATD2'nin eleman taşıma ve karşılaştırma karmaşıklığı temel alınan AT2 algoritmasıyla aynıdır (Eşitlik 3.2).

$$TAŞ[ATD2] = O(n + m + m\log m),\ KAR[ATD2] = O(n + m),\ m \le n \quad (3.2)$$

m = n = N/2 olması durumunda gerçekleşecek eleman taşıma ve karşılaştırma işlemlerinin karmaşıklığı Eşitlik 3.3'de verilmiştir.

$$TAŞ[ATD2] = O(N \log N) \text{ ve } KAR[ATD2] = O(N) \qquad (3.3)$$

ATD2 algoritmasının sıralı (sequential) hale getirilmesiyle geliştirilen ATY2 algoritmasında problemin iki alt probleme ayrıştırma sürecinde bir değişiklik yapılmamıştır. Farklılık alt problemlerden sadece birinin problem olarak ele alınmasından ortaya çıkmaktadır. Bu süreç bir döngü içinde derinlikteki alt problem bitinceye kadar devam eder. İşlemin sonucunda sıralanmamış olan kalan kesimde seri karşılaştırma işlemleriyle ilk alt problemdeki blok sınırları bulunarak aynı çözüm süreci uygulanmaktadır. Tüm dizin sıralı hale gelinceye kadar aynı işlemler devam etmektedir. Fakat derinlikteki hangi alt problemin yeni problem seçileceği ve bunun sonunda sıralanmamış kesimdeki hangi alt problemin bulunup ele alınacağı başlangıçtaki m ve n elemanlı problem bloklarının uzunluğuna göre belirlenmektedir. Buradaki hedef bölümleme işlemini büyük olan alt probleme uygulayarak her döngüde olabildiğince çok ve küçük boyutlu alt problemler yaratmaktır. Böylece yeni problemin sınırlarının bulunmasından kaynaklanan ek karşılaştırma işleri en aza indirgenmiş olunur.

ATY2 algoritmasında $m \leq n$ için alt problemlerden büyüğü sonda bulunmaktadır. Sondaki alt problemi problem alarak derinlemesine bölümleme yapan ve sıralanmamış kesimde sondaki ilk alt problemi alan çözüm süreci ATY2Son diye adlandırılmıştır. Diğer durumda ise alt problemlerden büyüğü başta ($m>n$) bulunmaktadır. Bu durumda baştaki alt problem yeni problem olarak alınmaktadır. Baştaki alt problemi problem alarak derinlemesine bölümleme yapan ve sıralanmamış kesimde baştaki ilk alt problemi alan çözüm süreci ATY2Bas diye adlandırılmıştır. ATY2Bas algoritmasının çözüm süreci Tablo 3.2'de gösterilmiştir.

Başlangıçta D=L[0..m] ve H=L[m+1..N-1] ,N=m+n Sıralanmamış Kesim ATY2Son için L[0..i] ve ATY2Bas için L[i..N-1] ,i∈ [0..N-1]	
ATY2Son m ≤ n	• Sıralanmamış Kesimde Eleman Varken o H bloğu ▪ Başlangıç ve bitiş indisleri hd=hh= i olarak ilklenir. ▪ L[hd] ≥ L[hd-1] olduğu sürece hd=hd - 1. ▪ L[hd..hh] kesimi olarak seçilir. o D bloğu ▪ Başlangıç ve bitiş indisleri dd=dh=hd - 1 olarak ilklenir. ▪ L[dd] ≥L[hd] olduğu sürece dd=dd-1. ▪ L[dd..dh] kesimi olarak seçilir o DH sırasızken ▪ ATD2 Ayrıştırma Süreci ▪ D=D_2 ve H=H_2
ATY2Bas m>n	• Sıralanmamış Kesimde Eleman Varken o D bloğu ▪ Başlangıç ve bitiş indisleri dd=dh= i olarak ilklenir. ▪ L[dh] ≤ L[dh+1] olduğu sürece dh=dh+1. ▪ L[dd..dh] kesimi olarak seçilir. o H bloğu ▪ Başlangıç ve bitiş indisleri hd=hh=dh+1 olarak ilklenir. ▪ L[hh] ≤ L[dh] olduğu sürece hh=hh+1. ▪ L[hd..hh] kesimi olarak seçilir. o DH sırasızken ▪ ATD2 Ayrıştırma Süreci ▪ D=D_1 ve H=H_1

Tablo 3.2 ATY2 Çözüm Süreci

ATD2 algoritmasının yerinde haline getirilmesiyle geliştirilen ATY2 algoritmasında ortaya çıkan tek ek maliyet blok sınırlarının

bulunması için gerçekleşen karşılaştırma işlemleridir.

Teorem:

Blok sınırlarının bulunması işlemi ortalama durumda $O(m\log(m/KESME))$ karşılaştırma maliyeti getirir.

İspat:

Başlangıçta ilk bloğun küçük olduğu durumda ($m \le n$), m ve kesme değerinin ikinin tam katı - $m = 2^a$ ve $KESME = 2^b$, $a,b \in N$ ve $a \ge b$- olduğunu varsayalım. Değiştirilecek eleman sayısı i ($0 \le i \le m$) ortalama olarak m/2'ye eşittir. Her seferinde küçük bloğun yarısı değiştirilecek olursa toplam derinlik *log(m/KESME)* kadar olur. k'nıncı (*k = 0..log(m/KESME) - 1*) derinlikte blok sınırlarını bulma işleminden dolayı gerçekleşen ilave karşılaştırma sayısı X_k: bir alt derinlikte (k+1) bulunan $m_{k+1}n_{k+1}$ kesimindeki yeni problemin ek karşılaştırma sayısı X_{k+1}, kalan $m_{k+1}m_{k+1}$ kesimindeki ek karşılaştırma sayısı Y_{k+1} ve bu kalan $2m_{k+1}$ boyutundaki kesimin sınırlarının bulunması için gerçekleştirilen $2m_{k+1=}m_k$ karşılaştırmalarının toplamıdır (Eşitlik 3.4).

$$X_k = X_{k+1} + Y_{k+1} + m_k \qquad (3.4)$$

Y_{k+1}, X_{k+1}'e ve m_k, $m/2^k$ 'ye eşittir. Eşitlik 3.4'de yapılan gerekli değişiklerle Eşitlik 3.5 elde edilmiştir.

$$X_k = 2X_{k+1} + m/2^k \qquad (3.5)$$

Son derinlikte küçük kesim kesme değerine eşit olacağından ek karşılaştırma , $X_{\log(m/KESME)} = 0$, gerçeklenmez. Eşitlik 3.5'deki X_{k+1}'i , Eşitlik 3.5 biçiminde değiştirirsek ve bu işlemi $X_{log(m/KESME)}$'ye kadar devam ettirirsek, X_0 yani başlangıç problemindeki ek karşılaştırma sayısı *mlog(m/KESME)*'ye eşit olur (Eşitlik 3.6).

$$X_0 = \sum_{j=1}^{\log(m/kesme)} m = m(\log m / KESME) \qquad (3.6)$$

Böylece ATY2 algoritmasında ortalama durumda gerçekleşen taşıma ve karşılaştırma işlemleri Eşitlik 3.7'deki gibi olmaktadır.

$$KAR[ATY2] = TAŞ[ATY2] = \mathrm{O}(n + m + m\log m) \qquad (3.7)$$

$m \cong n$ olduğu durumda ise bu işlem sayıları Eşitlik 3.7'de gösterilmiştir.

$$T[TAŞ] = T[KAR] = \mathrm{O}(N \log N) \qquad (3.8)$$

3.2 Yerinde Sürümün Paralel Ortamlar için Düzenlenmesi

ATY2 algoritmasının ortak belleğe (shared-mermory) sahip çok işlemcili platformlarda daha hızlı çalışabilmesi için her işlemciye ayrı bir alt problem atanması ve birleştirme işleminin işlemcilerdeki bağımsız problemler üzerinde paralel olarak gerçekleştirilmesi düşünülmüştür. Paralel ortam, Koşutzamanlı Okuma ve Dışlayan Yazma (Concurrent Read Exclusive Write, CREW) erişim modeline (Grama et al., 2003) uygun ve ikinin tam katı olan p tane - $p = 2^s, s$ bir Doğal Sayı- işlemcili olarak düşünülmüştür. $i, j \in \{0,1...,2^s - 1\}$ ve $i < j$ için P_i, P_j sırasıyla i ve j numaralı işlemcileri göstermektedir.

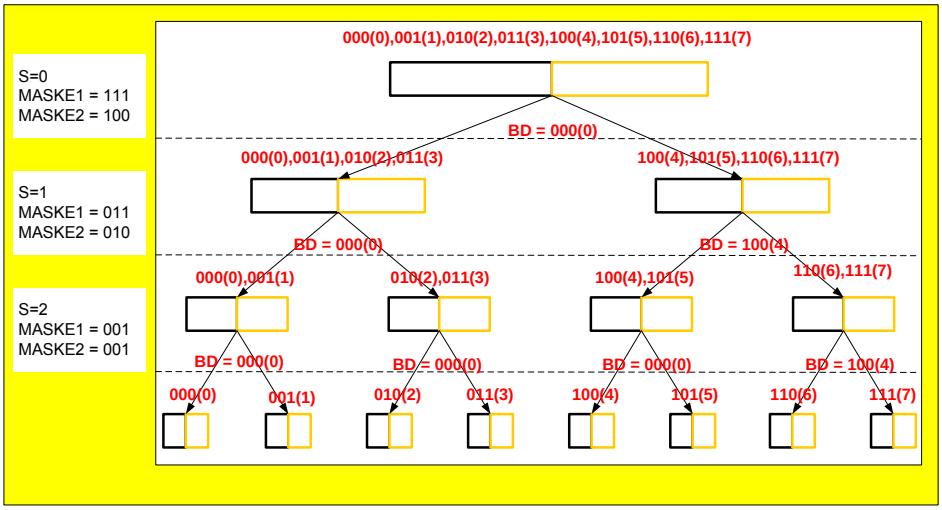
İş paylaşım süreci –her işlemciye bağımsız bir problem atanması-, s adımda gerçekleşen bölünme işlemlerinden oluşmaktadır. Birinci adımda, tüm işlemcilerde aynı ayrıştırma işlemi probleme uygulanarak değişecek blokların sınırları bulunur (Koşutzamanlı Okuma) ve sadece en küçük numaraya sahip olan işlemcide blok değiştirme işlemi (Dışlayan Yazma) gerçekleştirilir. Daha sonra işlemcilerden *p/2* tanesine baştaki alt problem, kalan *p/2* tanesine sondaki alt problem atanarak ikinci adıma

geçilir. Sonra ki adımlarda da aynı işlemler gerçekleştirilerek her P_i'ye özel D_iH_i alt problem atanmasıyla iş paylaşım süreci sonlanmaktadır. Elde edilen bağımsız iş parçacıklarından D_iH_i kesimindekinin en büyük eleman değeri, D_jH_j kesimindekinin en küçük eleman değerinden büyüktür. Bu alt problemlerin bulunduğu kesimlerin birleşimi, L[0..N-1] dizinine eşittir (Eşitlik 3.9).

$$\text{maks}(D_iH_i) \le \min(D_jH_j) \text{ ve } (\bigcup_{i=0}^{2^{s-1}} D_iH_i = L[0..N-1] \quad (3.9)$$

İş paylaşım sürecinin herhangi bir adımında, bir P_i'de blok değiştirme işleminin gerçekleşip gerçekleşmeyeceği ve alt problemlerden hangisinin atanacağının seçilmesi için maskeler (*maske₁* ve *maske₂*) kullanılmaktadır. Bu maske değerleriyle işlemci numaralarının "VE (AND)" işleminden (operator) geçirilmesi ile hesaplanan değere göre seçim kararı verilmektedir. *"maske₁* VE *i*" sıfıra eşit ise P_i'de blok değiştirme işlemi gerçekleştirilmektedir. *"maske₂* VE *i*" sıfıra eşit ise P_i'ye baştaki alt problem değilse sondaki alt problem atanmaktadır.

Şekil 3.1'de 8 işlemcili bir sistemdeki iş paylaşım süreci gösterilmiştir. İşlem üç adımda gerçekleştirilmektedir. Her adımdaki *maske₁*, *maske₂* değerleriyle işlemci numaraları ikili sayı sisteminde gösterilmiştir. Birinci adımda, "111" olan *maske₁* ile blok değiştirme (BD) işlemini P_0 gerçekleştirir ve *"100" olan maske₂* ile $\langle P_0, P_1, P_2, P_3 \rangle$'e baştaki , $\langle P_4, P_5, P_6, P_7 \rangle$'e sondaki alt problem atanır. Diğer adımlarda aynı seçimler gösterilen maske değerleriyle yapılarak 8 işlemciye bağımsız iş parçacıkları paylaştırılır. Her işlemci birleştirme işlemini bu problemler üzerinde gerçekleştirir.



Şekil 3.1 8 İşlemci için İş Paylaşım Diyagramı

ATY2 algoritmasının paralel ortamlar için düzenlenmesiyle elde edilen hızlanmanın (Eşitlik 3.10) büyüklüğü iş paylaşımının dengeli bir şekilde yapılmasıyla doğru orantılıdır. Hızlanmayı etkileyen diğer bir faktörde iş paylaşım sürecinin her adımında işlemcilerin bir kısmının blok değiştirme işlemini beklemek zorunluluğundan işlevsiz kalmasıdır.

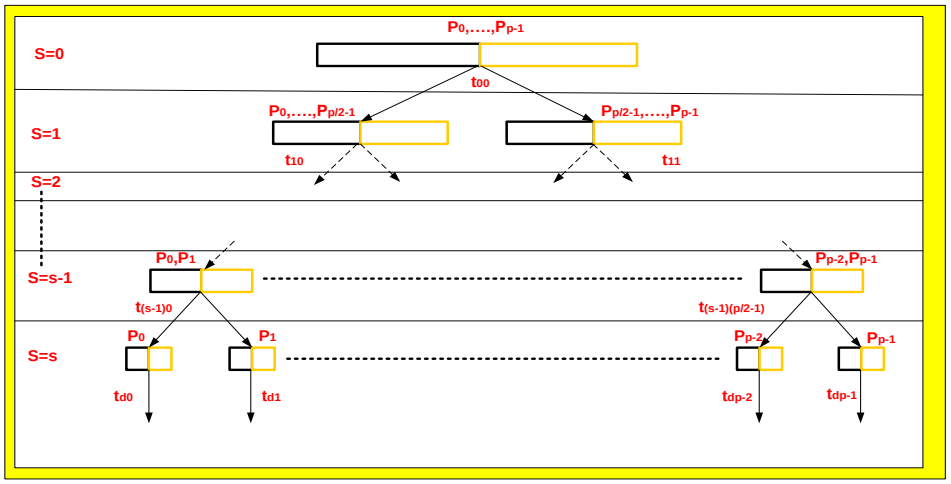
$$Hızlanma = T_{seri} / T_{paralel} \qquad \text{(Eşitlik 3.10)}$$

t_{SK} , $0 \le S \le s-1$ için S adım numaraları ve $K=(2^S.i)/p$, P_i'nin S'inci adımdaki işlem veya bekleme süresini, ve t_{si} ise bağımsız birleştirme süresini gösterecek olursa: P_i'nin toplam çalışma süresi $T[P_i] = \sum_{D=0}^{d} t_{DK}$ olur. İş paylaşım sürecindeki bu süreler Şekil 3.2'de gösterilmiştir. Örnek olarak 4 işlemciye sahip (p = 4) bir mimaride - $\langle P_0, P_1, P_2, P_3 \rangle$- iş paylaşım süreci 2 adımda (s = 2) gerçekleşmektedir. İlk adımda (S = 0 ve tüm işlemciler için K = 0) t_{00} süresi tüm işlemcilerin çalışmasında bulunmaktadır. İkinci ve son adımda (S = 1, $\langle P_0, P_1 \rangle$ için K

= 0 ve $\langle P_2, P_3 \rangle$ için K = 1), t_{10} süresi $\langle P_0, P_1 \rangle$ ve t_{10} süresi $\langle P_2, P_3 \rangle$ için ortaktır. Bu süreçten sonra her işlemci ($S = s = 2$) bağımsız problemlerde $\langle P_0, P_1, P_2, P_3 \rangle$ için $\langle t_{20}, t_{21}, t_{22}, t_{23} \rangle$ süre boyunca çalışmaktadır. Böylece işlemcilerin toplam çalışma süreleri: P_0 için $\langle t_{00} + t_{10} + t_{20} \rangle$, P_1 için $\langle t_{00} + t_{10} + t_{21} \rangle$, P_2 için $\langle t_{00} + t_{11} + t_{22} \rangle$ ve P_3 için $\langle t_{00} + t_{11} + t_{23} \rangle$ olur. Algoritmanın çalışma süresi bu sürelerin en büyüğüne eşittir.

Paralel çalışma süresi -$T_{paralel}$- işlemcilerden en uzun çalışanının süresine eşittir (Eşitlik 3.11).

$$T_{paralel} = maksimum(T[P_i]) \qquad (3.11)$$



Şekil 3.2 İşlemci Süreleri

Eşitlik 3.11'deki hızlanma böylece Eşitlik 3.12'deki gibi ifade edilir.

$$Hızlanma = T_{seri} / maksimum(T[P_i]) \qquad (3.11)$$

İş paylaşımı için ATD2 ayrıştırma sürecinin kullanılması ile alt

problemlerden birinin uzunluğu küçük parça (m) kadar olacağından, böyle bir bölünme m'in n'den belirli ölçüde küçük olduğu durumlarda işlemcilerden bazılarına büyük boyutlu problemler atanmasına sebep olacaktır. Bu sonuç dengesiz bir paylaşıma yol açacaktır. Bundan dolayı daha dengeli bir paylaşım sağlayacak ATD1 ayrıştırma sürecinin böyle durumlar için gerçekleştirilmesi yararlı olacaktır. İş paylaşımı sürecinde ATD1 ayrıştırma sürecini uygulayan paralel ATY2 algoritması ATP1 ve ATD2 ayrıştırma sürecini uygulayan paralel ATY2 algoritması ATP2 olarak adlandırılmıştır.

4 TESTLER VE SONUÇLARI

Üç farklı test ortamı hazırlanmıştır. Test ortamları ve test edilen algoritmalar C programlama diliyle Linux/Unix platformları için gerçekleştirilmiştir. Seri birleştirme ve sıralama gerçekleştirimleri 2 Ghz işlemciye sahip Linux platformunda ve paralel birleştirme benzetim gerçekleştirimleri ise aynı işlem gücüne sahip 8 bilgisayarlı Linux kümesinde (cluster) çalıştırılmıştır. Test edilen uygulamaların ele aldığı problemlerin giriş elemanları Sözde Rastgele Sayı Üreteci (Pseudo Random Number Generator, PRNG) ile üretilmiştir.

4.1 Seri Birleştirme Gerçekleştirimleri ve Testleri

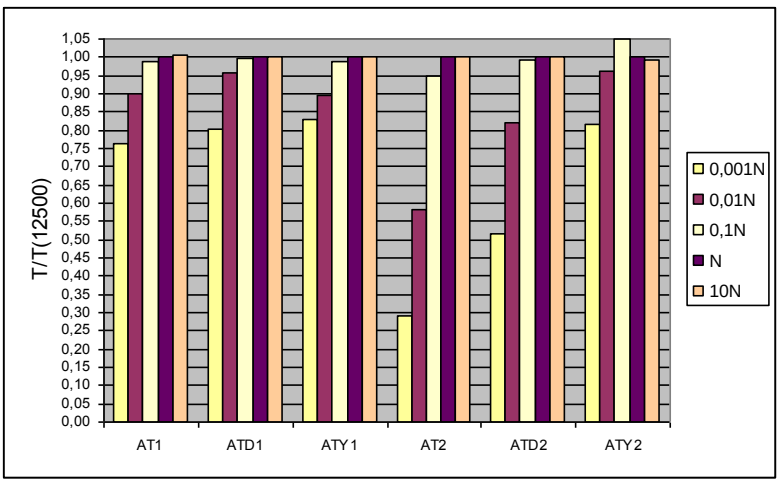
ATD2 ve ATY2 birleştirme algoritmaları ile bu algoritmaların kıyaslanabilmesi için AT1,ATD1,ATY1,AT2 algoritmaları C programlama dilinde gerçekleştirilmiştir (Ek 5). Gerçekleştirimlerin çalışma sürelerini normalleştirmek ve bu değer üzerinden karşılaştırmak için çalışma hızı yüksek olan standart bir birleştirme algoritması gerçekleştirilmiştir. O(N) ek bellek kullanarak, O(N) taşıma ve karşılaştırma işlemi ile birleştiren standart birleştirme algoritmasının (SBA) C diliyle gerçekleştirilmesi Ek 4.2'de verilmiştir. Giriş probleminin yapısı 3 kriter göz önünde bulundurularak oluşturulmuştur. Bunlar sırasıyla problemin uzunluğu(N), baştaki bloğun uzunluğunun dizin uzunluğuna oranı(m/N) ve dizin elemanlarının alabileceği değer kümesinin eleman sayısıdır. Farklı N uzunlukları için m'in olabileceği farklı uzunluklar ve elemanların seçildiği farklı değer sayısı Tablo 4.1'de gösterilmiştir. Dizin elemanlarının alabileceği değerler kümesinin eleman sayısı dizin uzunluğu (N) cinsinden <0,001N | 0,01N | 0,1N | N | 10N> -N = 10000 için <10 | 100 | 1000 | 10000 | 100000> - ifade edilmiştir. Kesme algoritmasının çalıştırılacağı değer (KESME) olarak 16 alınmıştır. Test

edilen gerçekleştirimlerin çalışma süreleri aynı kriterlere uygun 1000 giriş problemindeki çalışma sürelerinin ortalaması alınarak ölçülmüştür. Elde edilen süreler Ek 7.1'de verilmiştir.

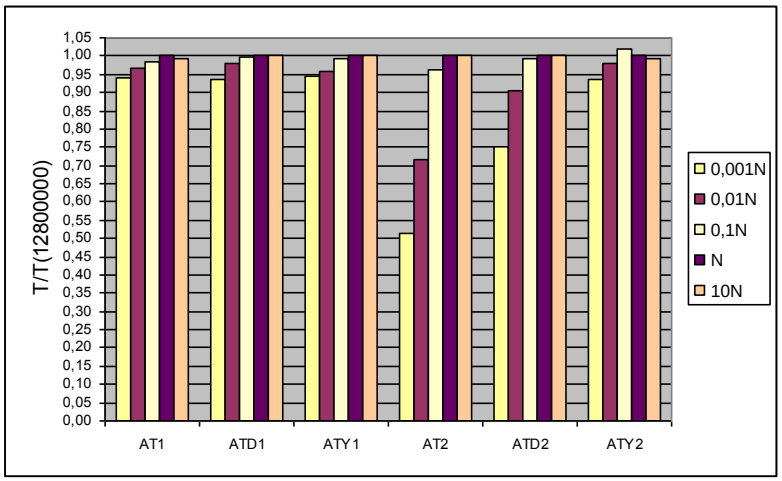
N: 12500; <25000 | 50000 | 100000 | 200000 | 400000| 800000 | 1600000| 3200000 | 6400000 | 12800000>
m/N: <0,1 | 0,25 | 0,5 | 0,75 | 0,9>
Değişik Eleman Değerleri: <0,001N | 0,01N | 0,1N | N | 10N>

Tablo 4.1 Problem Dizinindeki Kriterlerin Değer Kümeleri

Öncelikle değer kümesi değişiminin, çalışma sürelerine etkisi incelenmiştir. Her gerçekleştirimin aynı problem uzunluğunda ve blok uzunluğu oranındaki çalışma süreleri, değer kümesinin eleman sayısının problem uzunluğuna (N) eşit olduğu durumdaki çalışma süresine göre normalleştirilmiştir. m/N=0,1 iken N'in 12500 ve 12800000 değerleri için farklı değer kümelerindeki normalleştirilmiş süreler -sırasıyla değer kümesi eleman sayısı <0,001N | 0,01N | 0,1N | N | 10N>- sırasıyla Şekil 4.1 ve Şekil 4.2'de gösterilmiştir.

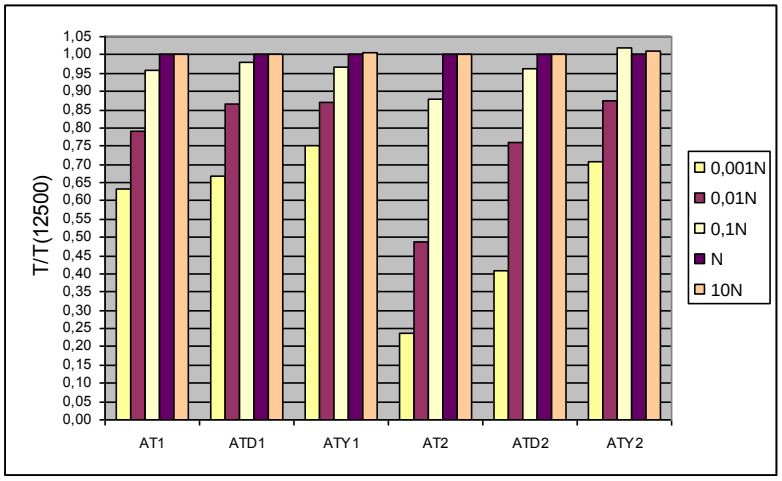


Şekil 4.1 m/N=0,1 ve N=12500 için Değer Kümesi Değişimi Etkisi

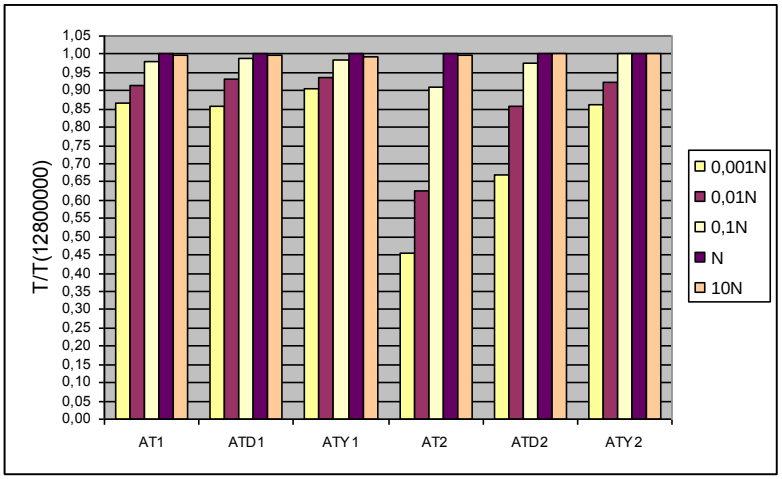


Şekil 4.2 m/N=0,1 ve N=12800000 için Değer Kümesi Değişimi Etkisi

m/N=0,25 iken N'in 12500 ve 12800000 değerleri için farklı değer kümelerindeki normalleştirilmiş süreler -sırasıyla değer kümesi eleman sayısı <0,001N | 0,01N | 0,1N | N | 10N>- sırasıyla Şekil 4.3 ve Şekil 4.4'de gösterilmiştir.

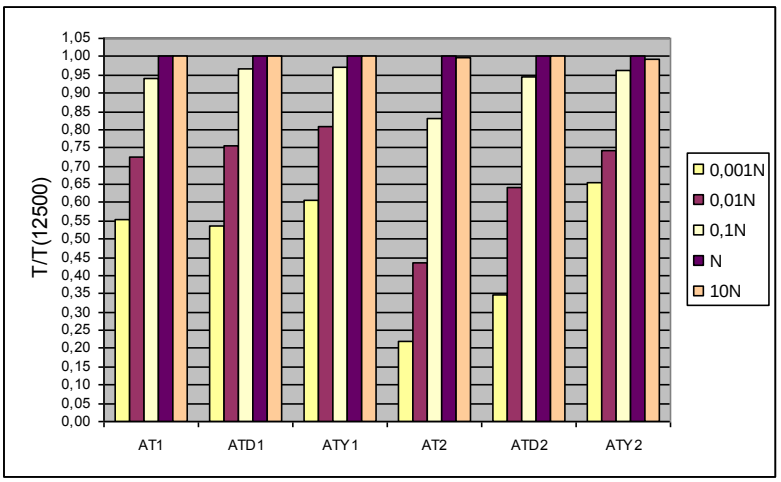


Şekil 4.3 m/N=0,25 ve N=12500 için Değer Kümesi Değişimi Etkisi

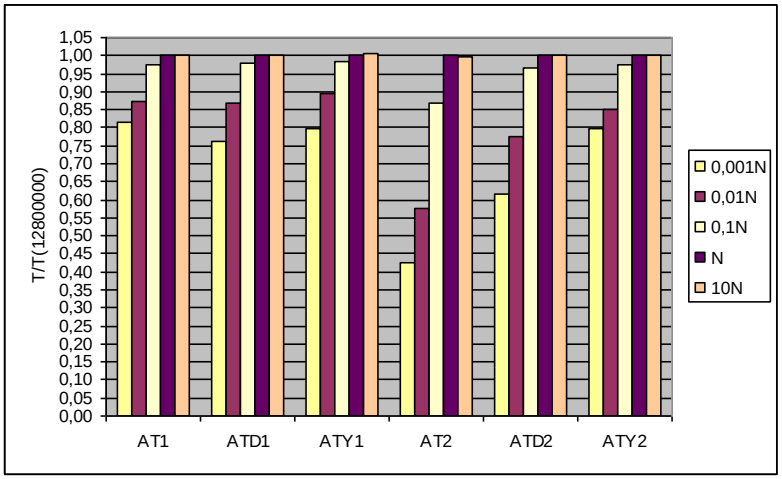


Şekil 4.4 m/N=0,25 ve N=12800000 için Değer Kümesi Değişimi Etkisi

m/N=0,5 iken N'in 12500 ve 12800000 değerleri için farklı değer kümelerindeki normalleştirilmiş süreler -sırasıyla değer kümesi eleman sayısı <0,001N | 0,01N | 0,1N | N | 10N>- sırasıyla Şekil 4.5 ve Şekil 4.6'da gösterilmiştir.

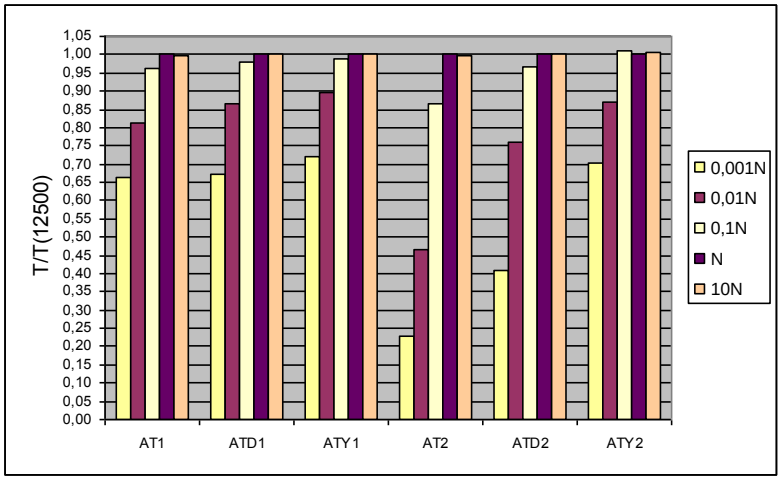


Şekil 4.5 m/N=0,5 ve N=12500 için Değer Kümesi Değişimi Etkisi

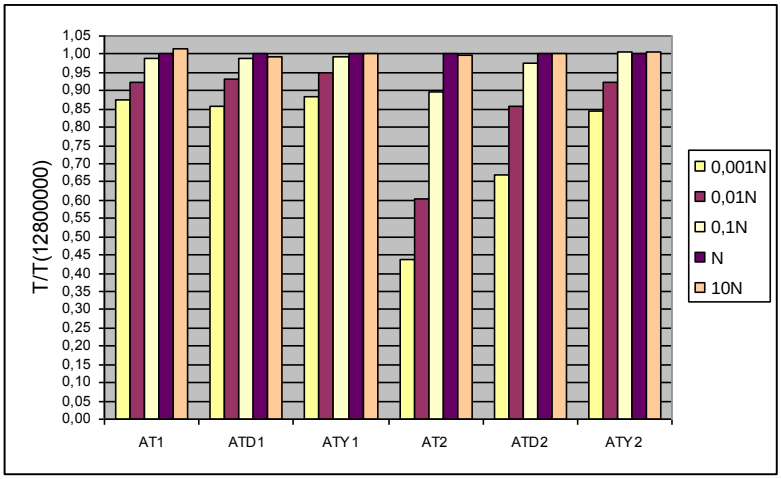


Şekil 4.6 m/N=0,5 ve N=12800000 için Değer Kümesi Değişimi Etkisi

m/N=0,75 iken N'in 12500 ve 12800000 değerleri için farklı değer kümelerindeki normalleştirilmiş süreler -sırasıyla değer kümesi eleman sayısı <0,001N | 0,01N | 0,1N | N | 10N>- sırasıyla Şekil 4.7 ve Şekil 4.8'de gösterilmiştir.

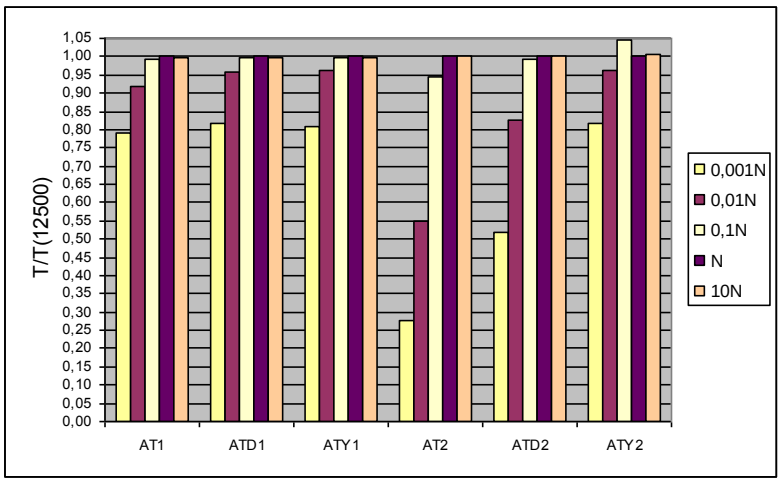


Şekil 4.7 m/N=0,75 ve N=12500 için Değer Kümesi Değişimi Etkisi

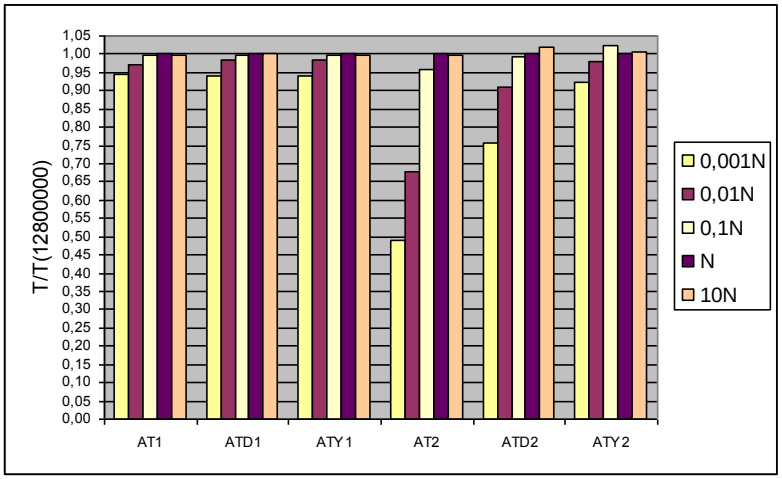


Şekil 4.8 m/N=0,75 ve N=12800000 için Değer Kümesi Değişimi Etkisi

m/N=0,9 iken N'in 12500 ve 12800000 değerleri için farklı değer kümelerindeki normalleştirilmiş süreler -sırasıyla değer kümesi eleman sayısı <0,001N | 0,01N | 0,1N | N | 10N>- sırasıyla Şekil 4.9 ve Şekil 4.10'da gösterilmiştir.



Şekil 4.9 m/N=0,9 ve N=12500 için Değer Kümesi Değişimi Etkisi



Şekil 4.10 m/N=0,9 ve N=12800000 için Değer Kümesi Değişimi Etkisi

Değer kümesi eleman sayısının N'den büyük olduğu durumlarda (testlerde 10N), çalışma süreleri N olduğu durumdakinden farklılık göstermemektedir. Çünkü normal dağılıma göre üretilen giriş problemi eleman değerlerinin aynı değer olması olasılığı çok küçülmektedir. Rastgele sıralı giriş problemleri için ATY2 dışında kalan gerçekleştirimlerin çalışma süreleri: m/N'in tüm oranlarında, sabit dizin uzunluğunda (N) değer kümesi eleman sayısı N'den küçüldükçe düşmektedir. Dizin uzunluğu arttıkça bu düşüş azalmaktadır. AT2'deki çalışma zamanı iyileşmesini gösteren düşüş oranı AT1'den daha fazla olup, bu özellik iyileştirilmiş versiyonlarında da (ATD1 ve ATD2) devam etmektedir.

Gerçekleştirimlerin N'den küçük olan değer kümelerindeki çalışma süreleri oranlarını sırasıyla ATBA<O[0.001N]|O[0.01N]|O[0.1N]> gösterecek olursak: N = 12500 ve m/N = 0,5 için (Şekil 4.5) süre oranları AT1<0,56|0,73|0,94>, AT2<0,22|0,43|0,83>, ATD1<0,54|0,76|0,94>, ATD2<0,34|0,65|0,94>, ATY1<0,60|0,81|0,97>; N=12800000 ve m/N = 0,5 için (Şekil 4.6) süre oranları AT1<0,81|0,87|0,97>, AT2<0,42|0,57|0,87>,

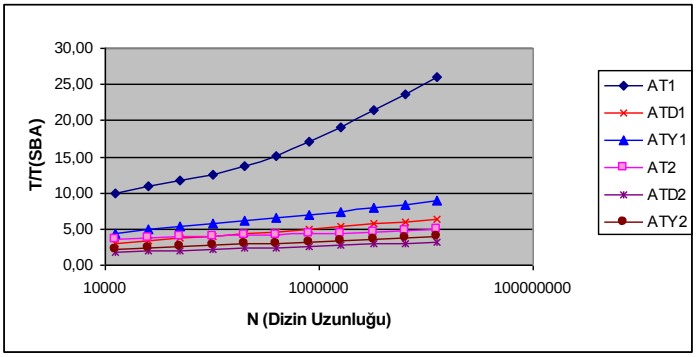
ATD1<0,76|0,87|0,98>, ATD2<0,62|0,78|0,96>, ATY1<0,80|0,90|0,98> olmaktadır. Değerlerden de görüleceği gibi AT2 ve ATD2 anahtar değerlerinin küçük bir kümeden seçilmesi durumunda ciddi bir performans artışı sergilemektedir.

Birbirlerinin eşleniği olan –küçük parça oranın aynı olduğu- m/n oranlarında (0,1 ile 0,9 ve 0,25 ile 0,75) gerçekleştirimlerin normalleştirilmiş süreleri birbirlerine çok yakın değerlerdedirler. N = 12500, m/N = 0,1 ve 0,9 için (Şekil 4.1 ve Şekil 4.9) süre oranları sırasıyla: AT1<0,76|0,90|0,99> ile AT1<0,79|0,92|0,99>, AT2<0,29|0,58|0,95> ile AT2<0,27|0,55|0,95>, ATD1<0,80|0,96|1,00> ile ATD1<0,82|0,96|1,00>, ATD2<0,52|0,82|0,99> ile ATD2<0,52|0,83|0,99>, ATY1<0,83|0,90|0,99> ile ATY1<0,82|0,96|1,00> olmaktadır. N = 12500, m/N = 0,25 ve 0,75 için (Şekil 4.3 ve Şekil 4.7) süre oranları sırasıyla: AT1<0,63|0,79|0,96> ile AT1<0,66|0,81|0,96>, AT2<0,24|0,49|0,88> ile AT2<0,23|0,47|0,87>, ATD1<0,67|0,86|0,98> ile ATD1<0,67|0,87|0,98>, ATD2<0,41|0,76|0,96> ile ATD2<0,41|0,76|0,96>, ATY1<0,75|0,87|0,97> ile ATY1<0,72|0,90|0,99> olmaktadır.

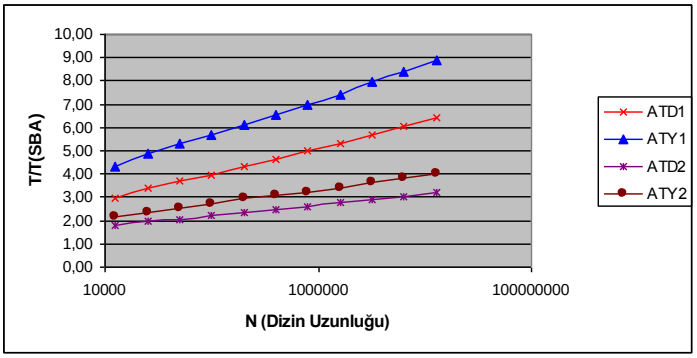
ATY2'nin değer kümesindeki değişimden etkilenmesi farklılık göstermektedir. Belli bir değer kümesi eleman sayısına kadar olan düşüş çalışma süresinde bir artışa sebep olmaktadır. N = 12500 ve m/N = 0,1 için (Şekil 4.1) çalışma süreleri oranları ATY2<0,82|0,96|1,05> ve N = 12800000 ve m/N = 0,1 için (Şekil 4.2) çalışma süreleri oranları ATY2<0,94|0,98|1,02> olarak değer kümesinin eleman sayısının dizin uzunluğunun %10'u (0,1N) olduğu durumda çalışma süresinde bir artışa sebep olmuştur. Bu değerden sonraki farklı eleman sayısındaki azalmalarda çalışma süresi düşmektedir.

Gerçekleştirilen algoritmalarının hızlarını birbirleriyle kıyaslamak amacıyla değişik eleman değeri sayısı N olarak alınarak farklı dizin uzunluklarındaki ve m/N oranlarındaki çalışma süreleri aynı kriterlerde

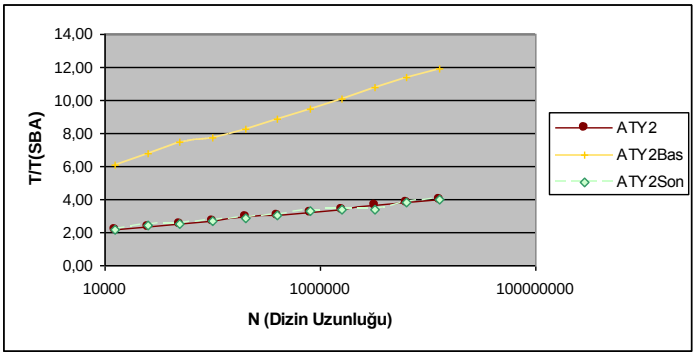
çalışan SBA gerçekleştirimin süreleri ile normalleştirilmiştir (Ek 7.2). m/N=0,1 için normalleştirilmiş sürelerinin karşılaştırmaları Şekil 4.11'de gösterilmiştir.



(a)



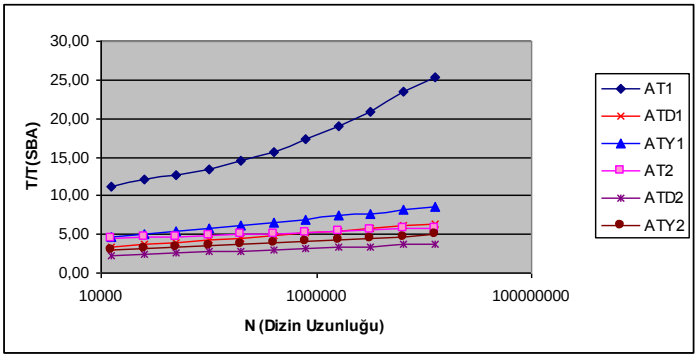
(b)



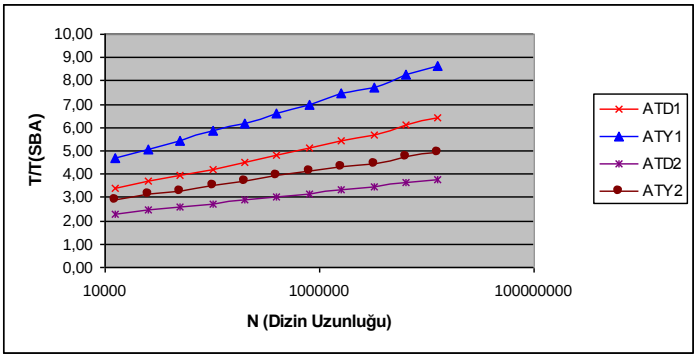
(c)

Şekil 4.11 m/N=0,1 için Süre Karşılaştırmaları

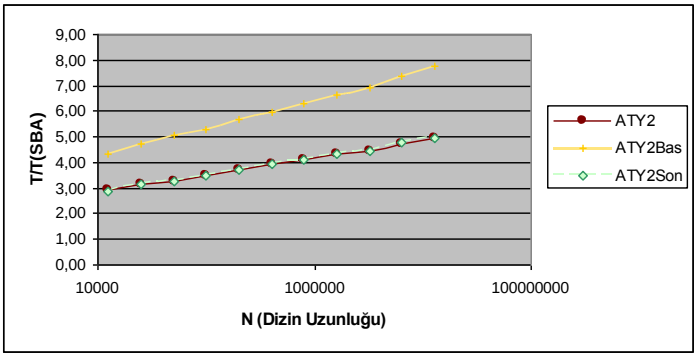
m/N=0,25 için normalleştirilmiş sürelerinin karşılaştırmaları Şekil 4.12'de gösterilmiştir.



(a)



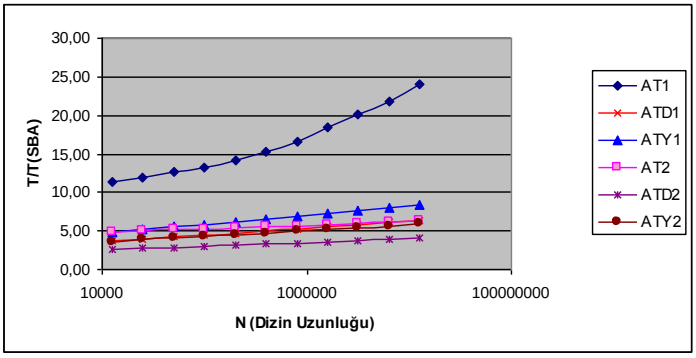
(b)



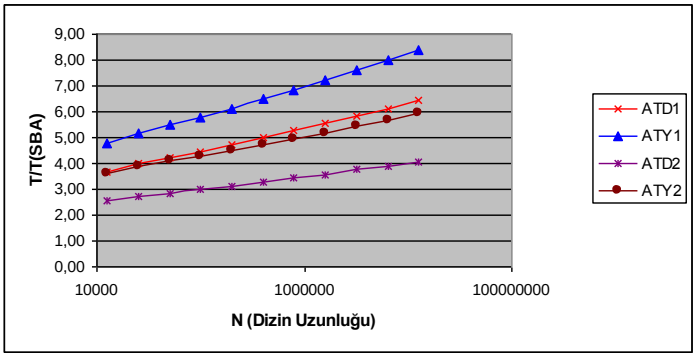
(c)

Şekil 4.12 m/N=0,25 için Süre Karşılaştırmaları

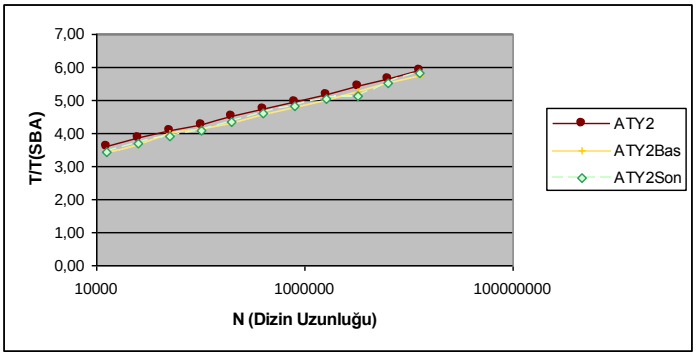
m/N=0,5 için normalleştirilmiş sürelerinin karşılaştırmaları Şekil 4.13'de gösterilmiştir.



(a)



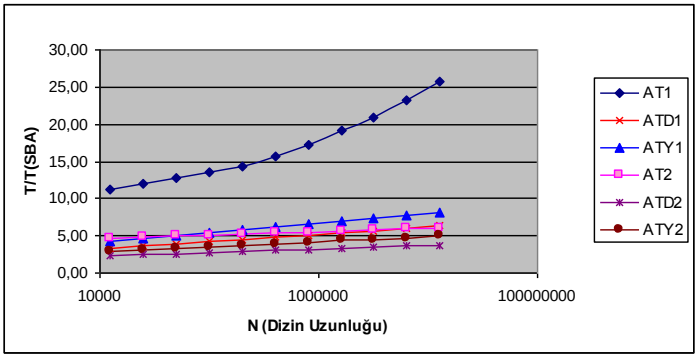
(b)



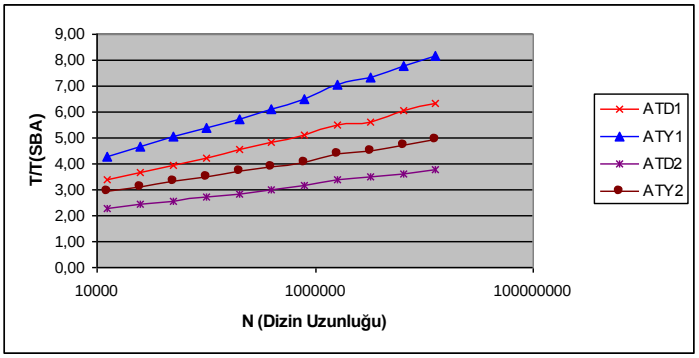
(c)

Şekil 4.13 m/N=0,5 için Süre Karşılaştırmaları

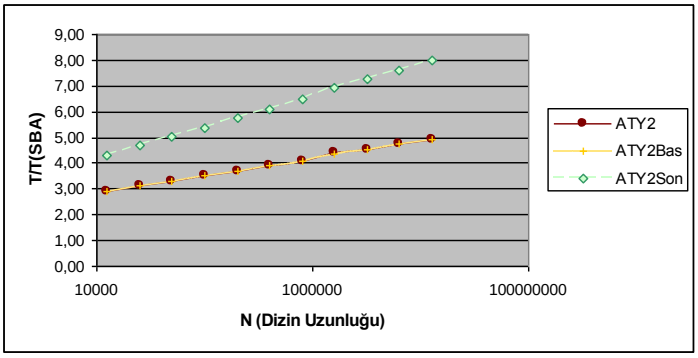
m/N=0,75 için normalleştirilmiş sürelerinin karşılaştırmaları Şekil 4.14'de gösterilmiştir.



(a)



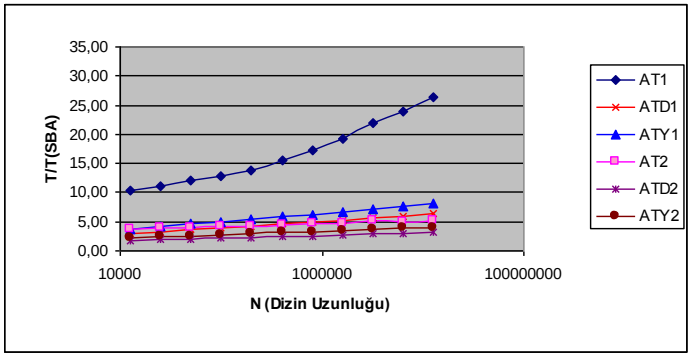
(b)



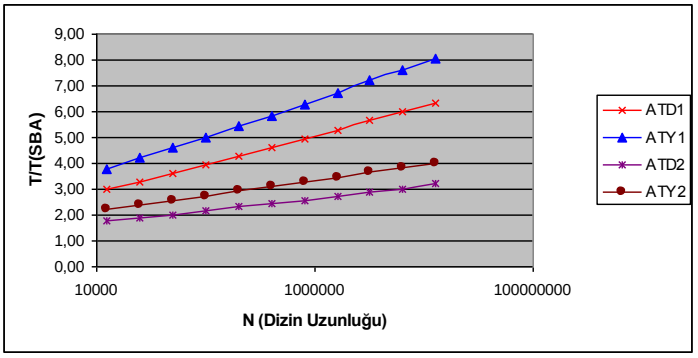
(c)

Şekil 4.14 m/N=0,75 için Süre Karşılaştırmaları

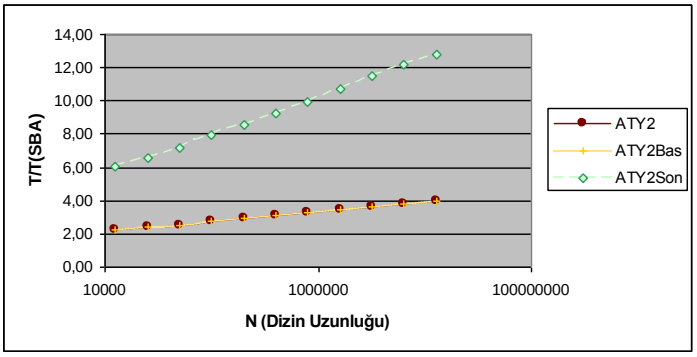
m/N=0,9 için normalleştirilmiş sürelerinin karşılaştırmaları Şekil 4.15'de gösterilmiştir.



(a)



(b)



(c)

Şekil 4.15 m/N=0,9 için Süre Karşılaştırmaları

Tüm m/N oranlarında ATBA gerçekleştirimlerin çalışma süreleri oranları dizin uzunluğunun artmasıyla artmaktadır (Şekil 4.11.a, Şekil 4.12.a, Şekil 4.13.a, Şekil 4.14.a, Şekil 4.15.a). Dizin uzunluğundaki artış; AT1 ve AT2'nin normalleştirilmiş değerlerinin sırasıyla yaklaşık olarak 10 - 25 ve 3,5 - 6,3 arasında, ATD1 ve ATD2 için sırasıyla yaklaşık 3 - 6,4 ve 1,8 - 4 arasında, ATY1 ve ATY2 için ise sırasıyla yaklaşık olarak 4,3 - 8,8 ve 2,2 - 6 arasında değişimine yol açmaktadır.

Rastgele sıralı giriş problemleri için bütün m/N oranlarında sağlanmış olmak üzere ATD1, ATD2, ATY1, ATY2 çalışma süreleri arasında (Şekil 4.11.b, Şekil 4.12.b, Şekil 4.13.b, Şekil 4.14.b, Şekil 4.15.b):

T[ATY1] > T[ATD1] > T[ATY2] > T[ATD2]

ilişkisi bulunmaktadır. Giriş probleminin küçük parçasının dizine oranının küçülmesiyle (|m/N - 0,5|) ATD1'in süresi ATY2'in süresine yaklaşmaktadır (Şekil 4.13.b).

Küçük parçanın başta bulunduğu durumlarda (m/N < 0,5) ATY2Son'un çalışma süresi ATY2Bas'tan daha kısa olmaktadır (Şekil 4.11.c, Şekil 4.12.c). Baştaki parçanın büyüklüğü arttıkça ATY2Bas ATY2Son'a yaklaşarak parçalar eşit olduğunda sürelerde eşitlenmektedir (Şekil 4.13.c). Baştaki parçanın büyük olduğu durumda ise ATY2Bas ATY2Son'dan daha hızlı çalışmaktadır (Şekil 4.14.c, Şekil 4.15.c). Bu çalışma hızlarına paralel olarak ATY2'nin çalışma süresi gerçekleştirimlerden hızlı olanına yaklaşmaktadır.

4.2 Sıralama Uygulamaları ve Testleri

Geliştirilen ATD2 ve ATY2, ile Dudzinski ve Dydek'in (ATD1ve ATY1) ayrıştırma tabanlı birleştirme algoritmalarının sıralama uygulamasında kullanımı, alttan yukarı (bottom-up, BU) yaklaşımla geliştirilmiş; iyileştirilmiş BirleştirSırala (Sedgewick, 1998) algoritmasının birleştirme evresine gömülerek gerçekleştirilmiştir. İyileştirme belli bir kesme değerinde(KESME2) veya küçük eleman sayılarındaki blokların sıralanmasında *InsertionSort* kullanılmasıdır. *Quicksort* için de ispatlanan bu iyileştirme BirleştirSırala (Mergesort) gerçekleştiriminin çalışma süresini %10-%15 azaltmaktadır(Sedgewick, 1998). Birleştirme evresinde ayrıştırma tabanlı birleştirme algoritması (ATBA) kullanan BirleştirSırala algoritmasının (msort_ATBA) C diliyle gerçekleştirimi Tablo 4.2'de verilmiştir.

```
void msort_ATBA(int L[],int dd,int hh){

     int i,m,j,k=1;

     j= KESME2;
     while( (j=j/2) !=0) k *=2;
     m= min(KESME2,k);

     for(i=dd;i<=hh-1;i += m)
          InsertionSort( L,i,i+min((hh-dd),(m-1)) );

     for (  ;m<=hh-dd;m=m+m)
          for(i=dd;i<=hh-m;i += m+m)
               BA(L,i,i+m-1,min(i+m+m-1,hh));
}
```

Tablo 4.2 BirleştirSırala'nın C diliyle Gerçekleştirimi

Sıralama uygulamalarının kıyaslanması için böl ve fethet (divide and conquer) modeline göre yukarıdan-aşağı (top-down, TD) yaklaşımla geliştirilmiş; iyileştirilmiş BirleştirSırala (Sedgewick, 1998) sıralama algoritması gerçekleştirilmiştir. Bu gerçekleştirim (msort_td) küçük blokların sıralanmasında *InsertionSort* kullanılması ve birleştirme evresinde yardımcı dizine kopyalama işleminin ortadan kaldırılması gibi iyileştirmeler içermektedir. C dilinde yazılan msort_td programı Ek 8.1'de verilmiştir.

Gerçekleştirilen BirleştirSırala uygulamalarının kararlı özelliğe sahip olması birleştirme evresinde çalıştırılan birleştirme algoritmasının kararlı olmasına bağlıdır(Sedgewick,1998). Ele alınan tüm birleştirme algoritmaları kararlı oldukları için sıralama uygulamaları da kararlıdır. TD yaklaşımı bölme evresinde özyinelemeden dolayı N uzunluğundaki bir dizi için O(log N) bellek karmaşıklığı yaratmaktadır. Ayrıştırma tabanlı birleştirme algoritmalarının BU yaklaşımlı sıralama algoritmasına gömülmesi ile birleştirme evresi dışında başka bellek kullanımının ortadan kalkması amaçlanmıştır. Böylece BU yaklaşımlı sıralama algoritmasının bellek karmaşıklığı ATBA'nın bellek karmaşıklığına- $O(ATBA_{bellek})$- eşit olur. msort_ATBA'nın taşıma ve karşılaştırma işlemleri bakımından karmaşıklığı birleştirme algoritması cinsinden Eşitlik 4.1'de verilmiştir.

$$T_{TAŞ}[msort\_ATBA] = O(T_{TAŞ}\lfloor ATBA \rfloor \log N)$$
$$T_{KAR}[msort\_ATBA] = O(T_{KAR}\lfloor ATBA \rfloor \log N) \quad \text{(Eşitlik 4.1)}$$

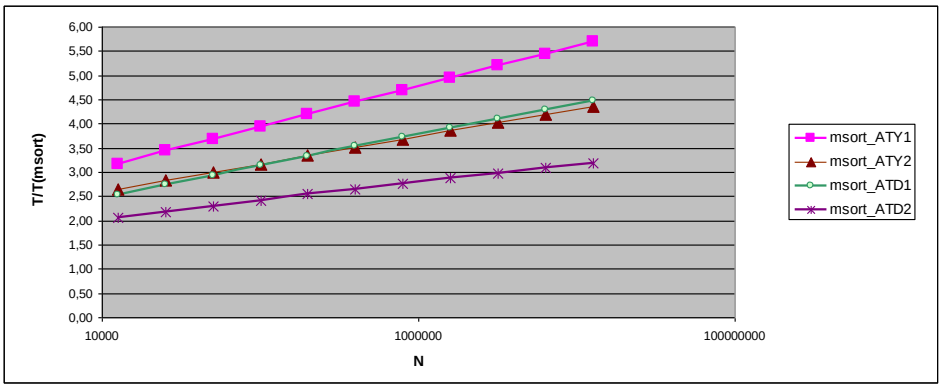
Eşitlik 3.2, 3.9, 2.11 ve 2.14'teki ATBA algoritmalarının karmaşıklıklarını Eşitlik 4.1'de yerine konulmalarıyla elde edilen sıralama algoritmalarının taşıma ve karşılaştırma karmaşıklıkları Tablo 4.3'te verilmiştir.

Algoritma	Bellek	Taşıma	Karşılaştırma
msort_ATD1	O(log N)	$O(N(\log N)^2)$	O(N log N))
msort_ATY1	O(1)	$O(N(\log N)^2)$	$O(N(\log N)^2)$
msort_ATD2	O(N)	$O(N(\log N)^2)$	O(N log N)
msort_ATY2	O(1)	$O(N(\log N)^2)$	$O(N(\log N)^2)$

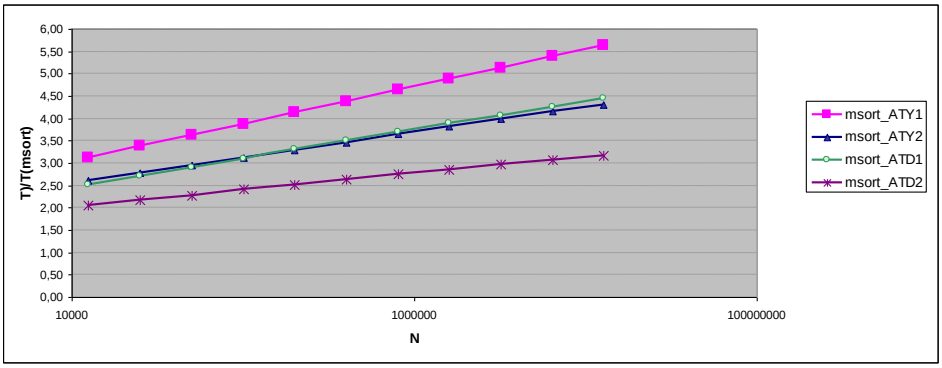
Tablo 4.3 Farklı Ayrıştırma Tabanlı Birleştirme Algoritmaları için BirleştirSırala Algoritmasının Karmaşıklıkları

İki farklı kesme değeri için gerçekleştirilen sıralama uygulamalarının testleri yapılmıştır. İlk kesme değeri (KESME1) birleştirme algoritmalarının BBA algoritması kullanması için gerekli olandır. İkincisi (KESME2) ise sıralama algoritmasında bulunan *Insertionsort* kullanması için gerekli olandır. Bu iki değerin (16,16), (16,32), (32,16) olduğu durumlarda değişik uzunluktaki dizinlerin sıralanması için ölçülen süreler Ek 8.2'de bulunmaktadır. Kesme değerlerinin (16,16), (16,32) ve (32,32) olduğu durumlarda birleştirme evresinde ATBA kullanan sıralama uygulamalarının ölçülen çalışma sürelerinin baz alınan msort_td uygulamasının (Ek 8.1) süreleri bölünmesiyle elde edilen normalleştirilmiş değerler (Ek 8.3) sırasıyla Şekil 4.16, 4.17 ve 4.18'de gösterilmiştir.

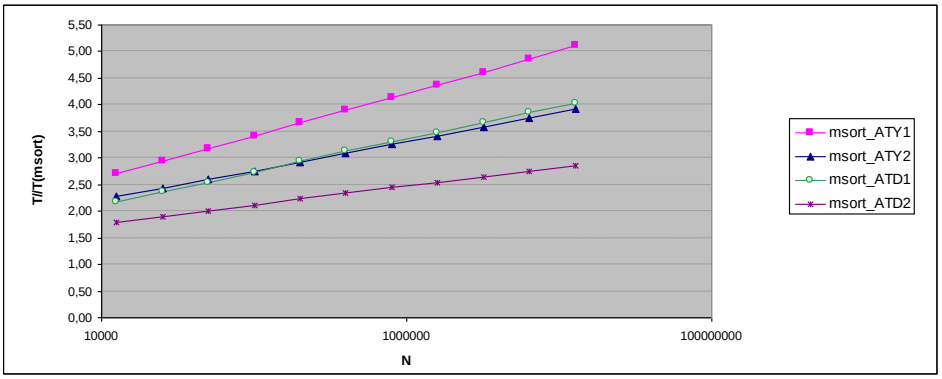
KESME2 değerinin 16'dan 32'ye çıkarılması (Şekil 4.16 ve 4.17) sıralama uygulamalarının çalışma sürelerinde belirli bir etki yapmamaktadır. Insertionsort ile gerçekleştirilen en iyi hızlanmayı sağlayan kesme değerinin 32'den küçük olması gerekmektedir.



Şekil 4.16 Kesme Değerleri(16,16) için Normalleştirilmiş Süreler



Şekil 4.17 Kesme Değerleri (16,32) için Normalleştirilmiş Süreler



Şekil 4.18 Kesme Değerleri (32,32) için Normalleştirilmiş Süreler

Tüm test durumları için:

$$msort\_ATD2 < msort\_ATY2 \cong msort\_ATD1 < msort\_ATY1$$

çalışma süresi sırası elde edilmiştir. ATBA ile geliştirilmiş BirleştirSırala (Mergesort) uygulamalarının baz alınan uygulamaya oranı dizin boyutuyla birlikte artmaktadır. KESME1 değeri 16 ve N=12500 olduğunda (Şekil 4.17) ATD1, ATD2, ATY2, ATY1 kullanan sıralama algoritmalarının çalışma sürelerinin baz alınan algoritmaya oranları sırasıyla <2,05|2,52|2,61|3,12> iken aynı değer 32'ye çıktığında(Şekil 4.18) bu oranlar <1,79|2,18|2,28|2,70> olarak yaklaşık %13'lük bir iyileşme sağlanmıştır. KESME1 değeri 16 ve N=12800000 olduğunda (Şekil 4.17) ATD1, ATD2, ATY2, ATY1 kullanan sıralama algoritmalarının oranları sırasıyla <3,17|4,45|4,32|5,65> iken aynı değer 32'ye çıktığında (Şekil 4.18) bu oranlar <2,85|4,03|3,91|5,11> olarak yaklaşık %10'luk bir iyileşme sağlanmıştır. Sonuç olarak KESME1 değerinin 2 katına çıkması tüm uygulamaların hızlarında yaklaşık %10-13 bir iyileştirme sağlamakla birlikte bu hızlanma dizin boyutunun artmasıyla azalmaktadır.

4.3 Paralel Birleştirme Gerçekleştirimleri Benzetimi

İleti Geçirme Arabirimi (Message Passing Interface, MPI), seri işlemcilerde paralel bir işi ileti aktarım yoluyla gerçekleştiren, yüksek performanslı paralel programlar yazılmasını mümkün kılan, Uygulama Programlama Arayüzü (Application Programming Interface, API) çağrımları içermektedir. Gerek iş istasyonlarından oluşan kümelerde, gerekse çok işlemcili platformlarda MPI çağrımlarını belirli standartlara (Gropp, 2003) göre yerine getiren uygulamalar geliştirilmiştir. Bu uygulamalar, MPI standartlarının yanında paralel iş için gerekli çalıştırma

ortamını da sağlamaktadırlar.

Ortak bellekli paralel ortamlar için geliştirilen ATP1 ve ATP2 algoritma hızlarını (Bkz. Eşitlik 3.10) ölçmek amacıyla LAM/MPI (The LAM/MPI Team, 2004) uygulamasının sağladığı ortamda ortak bellek benzetimi yapılarak algoritmalar, uygun bazı değişikliklerle gerçekleştirilmiştir. Paralel gerçekleştirimlerin çalıştırıldığı ortam aynı ve tek işlemci gücüne (2 Ghz) sahip 8 bilgisayarlı Linux kümesidir (Linux Cluster).

Ortak bellek benzetimi, bilgisayarlardan birinde üretilen birleştirme problemi dizininin diğer bilgisayarlara *MPI_Bcast( )* çağrımı kullanılarak dağıtılmasıyla sağlanmıştır. Her bilgisayarın -işlemcinin- kendi belleklerinde aynı diziye sahip olması ve onun üzerinde çalışmalarından dolayı algoritmalardaki *maske1* ile blok değiştirmeyi yapacak olan işlemcinin seçimi ortadan kaldırılmıştır. Bekleme süresi ile blok değiştirme süresi aynı olacağından ötürü değinilen değişikliğin en uzun çalıştırılan işlemcinin çalışma süresi üzerinde etkisi yoktur. En uzun çalışan bilgisayarın süresinin ölçümü, *MPI_Barrier( )* çağrımı ile birleştirme işlemine katılan her bilgisayarda çalışmaya başlama ve bitiş zamanlarının senkronizasyonunun sağlanmasıyla gerçekleştirilmektedir (Tablo 4.4). ATP1 ve ATP2 gerçekleştirimleri Ek 6'da bulunmaktadır.

Birleştirme işlemi bittikten sonra her bilgisayarda sıralanmış olarak bir alt-problem kesimi bulunmaktadır. *MPI_Gatherv( )* çağrımıyla bu kesimler, bilgisayarlardan birindeki dizinde toplanarak birleştirme işleminin doğru gerçekleştirilip gerçekleştirilmediği –dizinin sıralı olup/olmadığı- kontrol edilmiştir.

```
MPI_Barrier( );
gettimeofday(&t_ilk,0);
ATP1/ATP2( )
MPI_Barrier( );
gettimeofday(&t_son,0);
// T_paralel = t_son-t_ilk
```

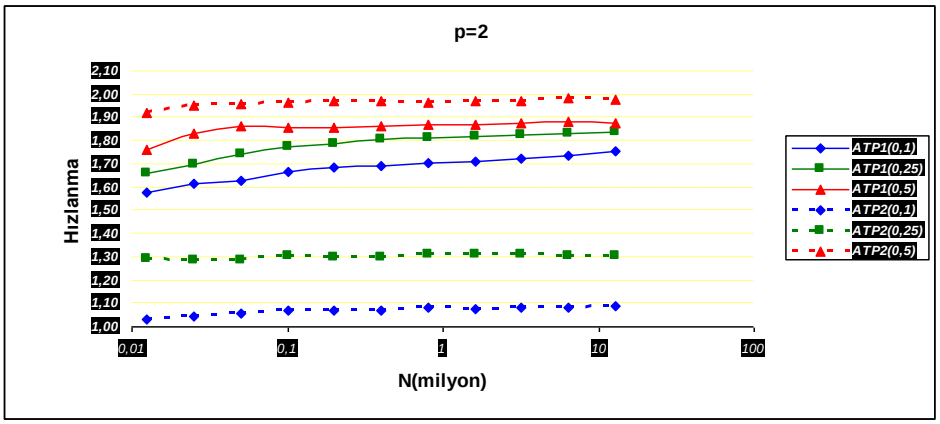
Tablo 4.4 Paralel Süre Ölçümleri

ATP1 ve ATP2 algoritmalarının gerçekleştirimlerinin çalışma süreleri, değer kümesi eleman sayısının dizin uzunluğuna eşit olduğu durum (N) için farklı blok oranlarında (m/N) -<0,1|0,25|0,5|0,75|0.9>- ve farklı işlemci sayılarında (p) -<2|4|8>- ölçülmüştür (Ek 9.1). Hızlanmanın hesaplanması (Bkz. Eşitlik 3.11) için aynı işlemci gücüne sahip bilgisayarda çalıştırılan ATY2'nin süreleri baz olarak kullanılmıştır (Ek 9.2). Bellek erişim hızları ve işlemci güçleri aynı olan ortamlarda paralel ve seri gerçekleştirimlerin çalıştırılması kıyaslamayı mümkün kılmaktadır. Eşitlik 4.2'ye (Grama et al., 2003) göre yapılan verim hesapları Ek 9.3'te verilmiştir.

$$Verim = Hızlanma / p \qquad \text{(Eşitlik 4.2)}$$

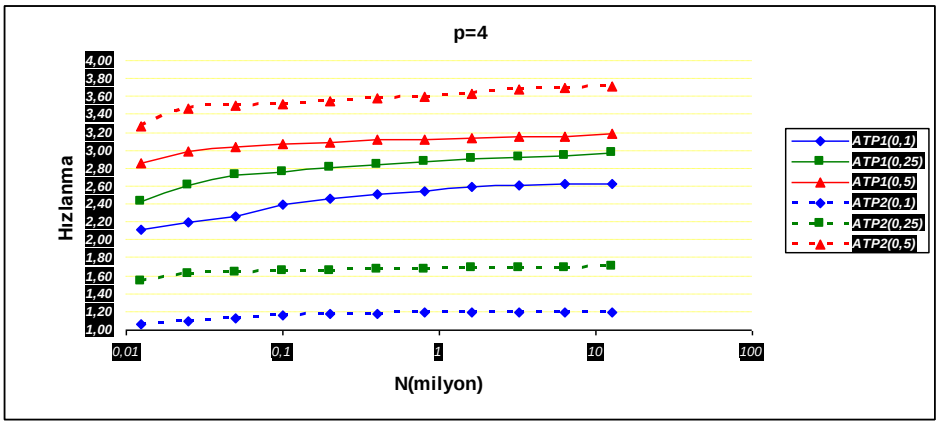
Tüm işlemci sayılarında ATP1 ve ATP2'nin hızlanma ve Verim (Eşitlik 4.2) değerleri birbirinin eşleniği olan uzunluk oranlarında (0,1 ile 0,9 ve 0,25 ile 0,75) yaklaşık olarak eşit hesaplanmıştır (Bkz. Ek 9.2 ve 9.3). Uzunluğu aynı olan küçük bloğun başta veya sonda bulunmasının hızlanma üzerinde belirli bir etkisi yoktur.

İşlemci sayısının 2 olduğu durumda, değişik m/N değerlerinde <0,1|0,25|0,5> algoritmaların farklı dizin uzunluklarına göre hızlanmaları Şekil 4.19'da gösterilmiştir.



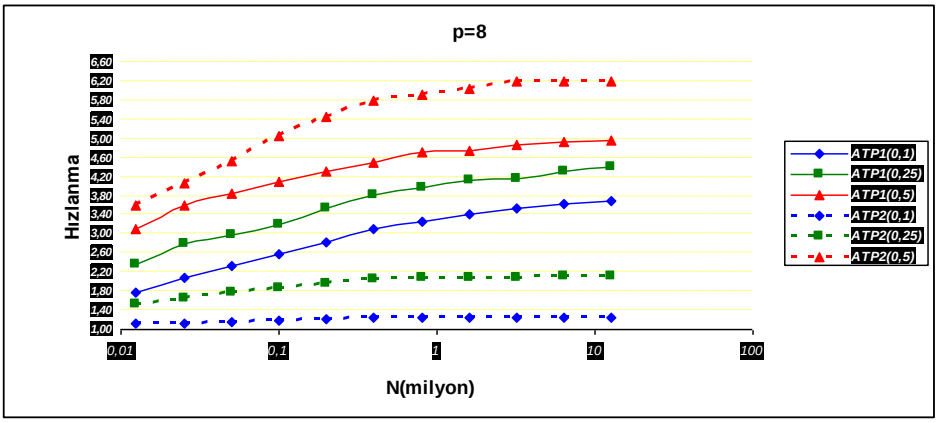
Şekil 4.19 Farklı m/N Oranlarında p=2 için Hızlanma vs N

İşlemci sayısının 4 olduğu durumda, değişik m/N değerlerinde <0,1|0,25|0,5> algoritmaların farklı dizin uzunluklarına göre hızlanmaları Şekil 4.20'de gösterilmiştir.



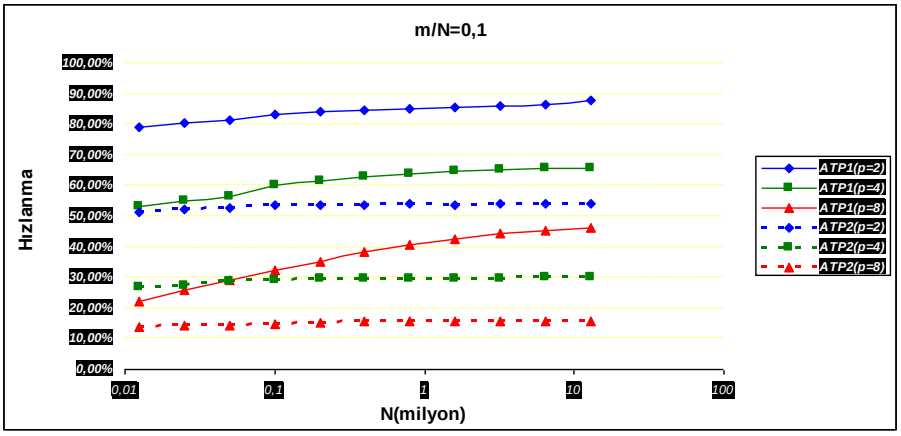
Şekil 4.20 Farklı m/N Oranlarında p=4 için Hızlanma vs N

İşlemci sayısının 8 olduğu durumda, değişik m/N değerlerinde <0,1|0,25|0,5> algoritmaların farklı dizin uzunluklarına göre hızlanmaları Şekil 4.21'de gösterilmiştir.



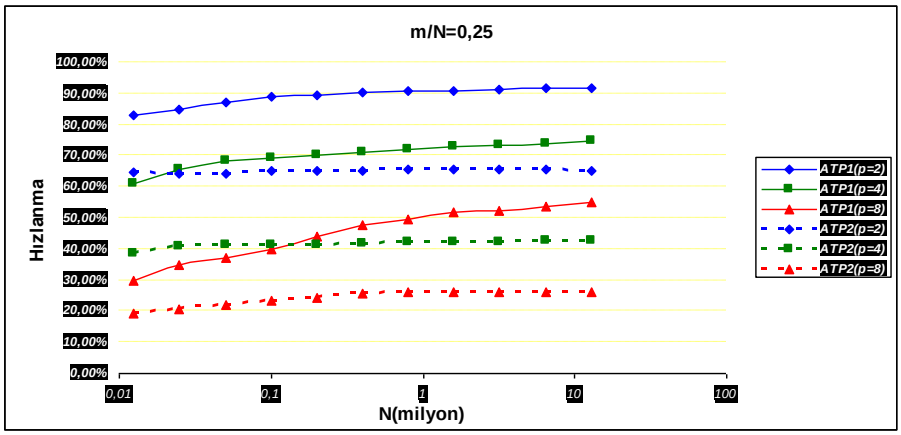
Şekil 4.21 Farklı m/N Oranlarında p=8 için Hızlanma vs N

m/N oranın 0,1 olduğu durumda, farklı işlemci sayılarında (p) <2|4|8> çalışan algoritmaların farklı dizin uzunluklarına göre hesaplanan verimleri Şekil 4.22'de gösterilmiştir.



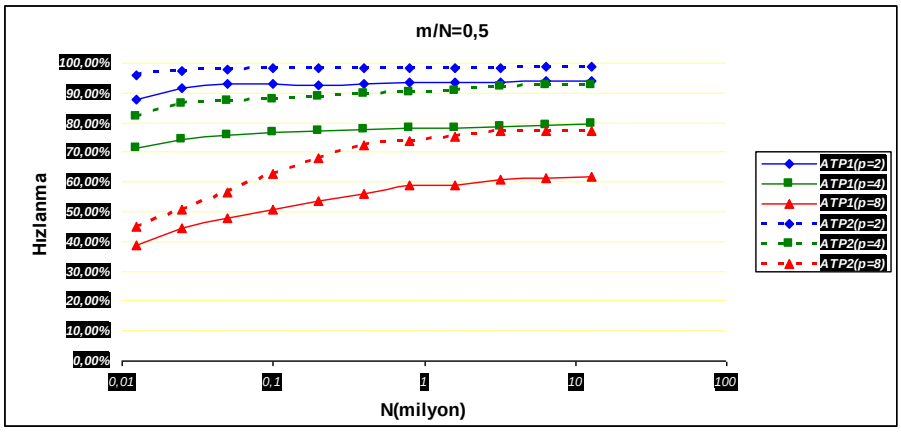
Şekil 4.22 Farklı İşlemci Sayılarında m/N =0,1 için Verim vs N

m/N oranın 0,25 olduğu durumda, farklı işlemci sayılarında (p) <2|4|8> çalışan algoritmaların farklı dizin uzunluklarına göre hesaplanan verimleri Şekil 4.23'te gösterilmiştir.



Şekil 4.23 Farklı İşlemci Sayılarında m/N =0,25 için Verim vs N

m/N oranın 0,5 olduğu durumda, farklı işlemci sayılarında (p) <2|4|8> çalışan algoritmaların farklı dizin uzunluklarına göre hesaplanan verimleri Şekil 4.24'te gösterilmiştir.



Şekil 4.24 Farklı İşlemci Sayılarında m/N =0,5 için Verim vs N

Sabit işlemci sayısında ve m/N oranında, ATP1 ve ATP2'nin hızlanma/verim değerleri N'in uzunluğunun 12500 ile 1280000 aralığında değişmesiyle artmaktadır. Değerlerdeki değişim aralıkları Tablo 4.5'de verilmiştir.

İşlemci Sayısı	m/N	Hızlanma Aralığı		Verim Aralığı	
		ATP1	ATP2	ATP1	ATP2
2	0,1	1,58-1,76	1,03-1,09	%79-88	%52-54
	0,25	1,66-1,84	1,29-1,31	%83-92	%64-66
	0,5	1,76-1,88	1,92-1,98	%88-94	%96-99
4	0,1	2,12-2,64	1,07-1,20	%53-65	%27-30
	0,25	2,44-2,98	1,54-1,71	%61-75	%39-43
	0,5	2,86-3,19	3,28-3,72	%71-80	%83-93
8	0,1	1,78-3,70	1,11-1,26	%22-46	%14-16
	0,25	2,37-4,39	1,54-2,10	%30-55	%19-26
	0,5	3,10-4,96	3,61-6,21	%38-62	%45-77

Tablo 4.5 Farklı Dizin Uzunluklarında Hızlanma/Verim Değişim Aralıkları

Paralel gerçekleştirimlerin ikisinde de sabit dizin uzunluğunda ve işlemci sayısında, m/N oranı arttıkça elde edilen hızlanma artmaktadır (Şekil 4.19, 4.20, 4.21). ATP1 küçük bloğunun oranın düşük olduğu değerlerde (0,1 ve 0,25) ATP2'den daha iyi bir paylaşım gerçekleştirerek daha hızlı çalışmaktadır. Küçük olan bloğun oranı arttıkça ATP2'in hızlanmasındaki artış oranı ATP1'den fazla olmaktadır.p=(2, 4, 8) için: m/N'in 0,1'den 0,25'e çıkması ile yaklaşık olarak ATP1'de (%5, %12, %20) hızlanma artışı sağlanırken, ATP2'de (%20, %40, %65) sağlanmaktadır; m/N'in 0,25'den 0,5'e çıkması ile yaklaşık olarak ATP1'de (%2, %8, %17) sağlanırken ATP2'de (%50, %215, %290) sağlanmaktadır. Blok uzunluklarının yakın olduğu durumda ($m \approx n$), ATP2 daha yüksek hızlanma/verim değerleriyle ATP1'den daha dengeli bir iş paylaşımı gerçekleştirir.

5 SONUÇ

Bu çalışmada geliştirilen ve kıyaslanma için ele alınan kararlı özelliğe sahip ayrıştırma tabanlı birleştirme algoritmalarının tamamı C programlama diliyle modüler olarak gerçekleştirilmiştir. Böylece birleştirme işlemine gereksinim duyan uygulamalar tarafından kolayca kullanılabilirliği sağlanmıştır. Ayrıca seri ayrıştırma tabanlı birleştirme algoritmalarının birleştirme evresine gömülmesiyle oluşturulan seri BirleştirSırala sıralama uygulamalarında, yine C programlama dili kullanılmıştır.

Algoritmalarda kullanılan blok değiştirme ve ikili arama temel tekniklerinin yazılımları bağımsız modüller olarak tasarlanmıştır. Bu çalışma ile; gerçekleştirilen algoritmaların iyileştirilmesi ve değişik yaklaşımlar ile geliştirilen yeni birleştirme algoritmalarının yazılıma aktarımları kolaylaştırılarak gelecek çalışmalar için bir kaynak oluşturulmuştur.

Hazırlanan 3 ayrı test ortamında –seri ATBA, seri BirleştirSırala (MergeSort) uygulamaları, paralel ATBA benzetimi için- her dizin uzunluğunda aynı parametrelere göre 1000 farklı giriş problemi üretilmiştir. Test süreleri bu giriş problemlerindeki toplam sürelerin ortalaması olarak hesaplanmıştır. Bu çalışmalara ait test sonuçları sunulmuş ve değerlendirilmiştir (Bkz. Bölüm 4).

Bu çalışma sırasında gerçekleştirilen tüm birleştirme, sıralama algoritmaları ve test programları Linux işletim sistemine sahip 2 Ghz işlemcili iş istasyonlarında çalıştırılmıştır. Paralel ortam olarak ileti geçirme prensibine dayanan MPI kütüphanesiyle sağlanan aynı özelliklere sahip 8 iş istasyonlu küme kullanılmıştır.

Temel alınan ayrıştırma tabanlı birleştirme algoritmalarında -AT1, AT2- yapılan değişiklikler -ATD1 ve ATD2- verimliliği önemli ölçüde

iyileştirmiştir. Bunun yanında, iyileştirilmiş algoritmaların O(1) bellek alanı kullanan yerinde uyarlamaları –ATY1 ve ATY2- tanımlanmıştır. Bu yerinde uyarlamalar, verimliliklerinde yeni problem sınırlarının bulunmasından ötürü bir düşme olmasına karşılık hala temel algoritmalara (AT1 ve AT2) göre hızlı çalışmaktadırlar. Birebir karşılaştırıldıklarında; AT2 ve türevlerinin performanslarının, AT1 ve türevlerinden büyük ölçüde yüksek olduğu sonucu elde edilmiştir. AT2 türevlerinin AT1 türevlerine üstünlüğü içine gömüldükleri BirleştirSırala sıralama uygulamalarında da devam etmiştir. Aşağıdan yukarı (BU) yaklaşımla gerçekleştirilen BirleştirSırala sıralama uygulaması seçilerek, birleştirme evresine yerinde uyarlamaların gömülmesiyle ortalama zamanda $O(N^2)$'in altında atama -O(NlogN)- ve karşılaştırma - $O(N\log^2 N)$- işlemi gerçekleştiren yerinde ve kararlı sıralama algoritmaları oluşturulmuştur.

Bu çalışma sırasında karşılaşılan en büyük problem paralel algoritmaların gerçekleştirilmesi ve testleri için gerekli olan ortak bellekli çok işlemcili bir platformun bulunamamasıdır. Bu yüzden gerçekleştirimler LAM/MPI ile oluşturulan paralel çalışma ortamında ortak bellek benzetimine uyarlanarak yapılmıştır. Gerçekleştirilen testler sonucunda AT1 tabanlı iş paylaşımının AT2 tabanlıdan giriş problemindeki blok oranları küçüldükçe daha dengeli olduğu ortaya çıkmıştır. Blok oranlarının birbirine yakın olduğu durumda ($m \approx n$), AT2 tabanlı iş paylaşımıyla daha yüksek bir hızlanma/verim elde edilmiştir. Bundan dolayı AT2'nin eşit uzunluktaki blokların birleştirildiği paralel BirleştirSırala sıralama uygulamalarında daha verimli çalışması beklenmektedir.

İleriye yönelik olarak geliştirilen ATBA algoritmalarının farklı özelliklere sahip veri kümeleri ve dağılımlara göre üretilen giriş problemleri üzerinde test edilmesi düşünülmektedir. Böylece çıkan

sonuçlara göre değişik özellikteki girişler için algoritmaların iyileştirilmesi mümkün olacaktır. Yapılacak bu uyarlamalarla en iyi çalışma performansları elde edilecektir. Bu çalışmada geliştirilen paralel birleştirme algoritmalarıyla birlikte kıyaslama içinde Gavril'in (Gavril, 1975) ve Baghavathi ile arkadaşlarının (Baghavathi et al., 1992) paralel birleştirme algoritmaları gibi değişik algoritmaların, çok işlemcili platformlar için gerçekleştirimlerinin ve testlerinin yapılması sonucunda daha doğru sonuçlar elde etmek mümkün olabilir. Ayrıca gerçekleştirilecek paralel algoritmaların paralel Mergesort sıralama uygulamasına gömülmesi ve performanslarının incelenmesi de diğer bir çalışma alanı olabilmektedir.

Sonuç olarak bu çalışma, gerek birleştirme/sıralama problemin bilgisayar bilimlerinde önemli bir alan olması gerekse ortaya konulan algoritmaların geliştirilmeye açık olmasından dolayı bundan sonraki yıllarda yapılacak çalışmalar için de kaynak niteliğindedir.

KAYNAKLAR DİZİNİ

Bhagavathi, D., Denny, W.M., Grosch, C., Looges, P.J., and Olariu, S., 1992, Sorting and Merging on the DAP, *ACM 30th Annual Southeast Conference*, 93-99p.

Chen, J., 2003, Optimizing Stable In-Place Merging, *Theoretical Computer Science*, 302: 191-210.

Dalkılıç, M.E., 2002 (Unpublished), Simple Recursive Stable In-Place Merge.

Dudzinski, K., and Dydek, A., 1981, On a Stable Storage Merging Algorithm, *Information Processing Letters*, 12: 5-8.

Dvorak, S. and Durian, B. ,1988, Merging by Decomposition Revisited, *The Computer Journal*, 31:279-282..

Estivill-Castro, V., and Wood, D.,1992, A Survey of Adaptive Sorting Algorithms, *ACM Computing Surveys*, Vol.24, No.4: 441-476.

Gavril, F., 1975, Merging with Parallel Processors, *Communications of ACM*, Vol.18, No.10: 588-591.

Geffert, V., Katajainen, J., and Pasanen, T., 2000, Asymptotically Efficient In-Place Merging, *Theoretical Computer Science*, 237: 159-181.

Grama, A., Gupta, A., Karypis, G., and Kumar, V., 2003, Introduction to Parallel Programming, 2nd Edition, Addison Wesley , 856p.

KAYNAKLAR DİZİNİ (DEVAM)

Gropp, W., 2003, Tutorial on MPI: Message Passing Interface, Argonne National Laboratory: Mathematics and Computer Science Division, 155p, *http://www-unix.mcs.anl.gov/mpi/tutorial/gropp/talk.html*

Hwang, F.K., and Lin, S., 1972, A Simple Algorithm for Merging Two Disjoint Linearly Ordered Sets, *SIAM Journal on Computing*, 1: 31-39.

Huang, B-C, 1992, Fast Stable Merging and Sorting in Constant Extra Space, *The Computer Journal*, 35: 643-650.

Knuth, D.E., 1973, The Art of Computer Programming, Vol. 3: Sorting and Searching, Addison Wesley, Reading, MA, 723p.

Pardo, L.T., 1977, Stable Sorting and Merging with Optimal Space and Time Bounds, *SIAM Journal on Computing*, 6: 351-372.

Sedgewick, R., 1998, Algorithms in C, 3rd Edition, Addison Wesley, 720p.

Symvonis, A., 1995, Optimal Stable Merging, *Computer Journal*, 38: 681-690.

The LAM/MPI Team, 2004, LAM/MPI User's Guide Version 7.0.6 , Pervasive Technology Labs, Indiana University, http://www.lam-mpi.org/download/files/7.0.6-user.pdf

Xiao, L., Zhang, X., and Kubricht, S.A., 2000, Improving Memory Performance of Sorting Algorithms, *ACM Journal on Experimental Algorithmics*, 5: 1-21.

EKLER

Ek 1 AT2 Karmaşıklık Analizleri

Ek 2 Blok Değiştirme Teknikleri Kodları

Ek 3 İkili Arama Teknikleri Kodları

Ek 4 Basit ve Standart Birleştirme Algoritmaları Kodları

Ek 5 Ayrıştırma Tabanlı Algoritmaların Kodları

Ek 6 Paralel Algoritmalarının Benzetim Kodları

Ek 7 Seri Birleştirme Gerçekleştirimleri Test Sonuçları

Ek 8 Sıralama Uygulamaları Test Sonuçları

Ek 9 Paralel Algoritmaların Benzetim Test Sonuçları

Ek 10 Türkçe-İngilizce Terimler Sözlüğü

Ek 1 AT2 Karmaşıklık Analizleri

i : özyineleme düzeyi.

j: alt problem numarası , $j = [1..2^i]$.

$(m,n)_{ij}$: i'inci düzeydeki j'inci alt problem.

$m_{ij} = \min [(m,n)_{ij}]$ ve $n_{ij} = \max[(m,n)_{ij}]$.

d_{ij} : $(m,n)_{ij}$ de değiştirilecek eleman sayısı.

k_{ij} : $(m,n)_{ij}$ de gerçekleşen karşılaştırma sayısı.

Başlangıçta –i=0, j=1- $m_{01} = m \le n = n_{01}$;

Durum 1: Küçük parçanın tamamı değiştirilecek ($d_{ij} = m_{ij}$) olan alt problemlerde toplam değiştirilecek eleman sayısı toplam dizin uzunluğunu geçemez:

$$\sum_1 d_{ij} \le m + n$$

Toplam karşılaştırma sayısı :

$$\sum_1 k_{ij} = \sum_1 \log d_{ij} \le \sum_1 d_{ij} \le m + n$$

Durum 2: $m_{ij} > 1$iken $k_{ij} \in \lfloor 1..m_{ij} - 1 \rfloor$ için değiştirilecek eleman sayısının büyütülmesi (maximize) büyük tarafta oluşturulacak (n_{ij}) yeni küçük bloğun büyütülmesiyle oluşmaktadır.

a) $n_{ij} \ge 2(m_{ij} - 1)$ iken değiştirilecek eleman sayısı $d_{ij} = m_{ij} - 1$ve toplamı:

$$\sum_{2.a} d_{ij} = \sum_{2.a} m_{ij} - 1 \le n$$

Karşılaştırma sayısı $\log m_{ij} = \log(d_{ij} + 1)$ ve toplamı:

$$\sum_{2.a} k_{ij} = \sum_{2.a} \log m_{ij} = \sum_{2.a} \log(d_{ij} + 1) \le \sum_{2.a} (d_{ij} + 1) = \sum_{2.a} d_{ij} + \sum_{2.a} 1 \le n + m$$

$\sum_{2.a} 1 \leq$ m : bu durumu sağlayacak alt problemlerin toplam sayısı başlangıçtaki küçük parça uzunluğunu (m) geçemez.

b) $n_{ij} < 2(m_{ij} - 1)$ iken $n'_{ij} = m'_{ij} = 2m_{ij}$ olan herhangi bir problemden daha az sayıda karşılaştırma ve değiştirilecek elemana sahiptir.

Yeni tanımlanan problemde büyütülme yarı eleman değişikliğiyle yani bloktaki elemanların yarısının değiştirilmesiyle olur.

$d_{ij} \leq d'_{ij} = m_{ij}$ ve $k_{ij} \leq k'_{ij} = \log 2m_{ij}$

herhangi bir $m_{ij} < m$ olacağından, adım sayısı log 2m için toplam değiştirme:

$$\sum_{2.b} d_{ij} \leq \sum_{2.b} d'_{ij} \leq m \log 2m$$

ve S=log 2m için karşılaştırma sayısı toplamı:

$$\sum_{2.b} k_{ij} \leq \sum_{2.b} k'_{ij} = \sum_{t=1}^{S} 2^t (S - t + 1) \cong 4m - 2$$

Böylece taşıma sayısı ve karşılaştırma karmaşıklıkları sırasıyla:

$$\#\text{TAŞ} \quad \leq 3 \left[\sum_{1} d_{ij} + \sum_{2.a} d_{ij} + \sum_{2.b} d_{ij} \right]$$

$$\leq 3 \left[n + m + n + m \log 2m \right] = \boldsymbol{O(m+n+mlogm)}$$

$$\#\text{KAR} \quad \leq \left[\sum_{1} k_{ij} + \sum_{2.a} k_{ij} + \sum_{2.b} k_{ij} \right]$$

$$\leq \left[m + n + m + n + \mathrm{O}(m) \right] = \boldsymbol{O(m+n)}$$

Ek 2 Blok Değiştirme Teknikleri Kodları

2.1 Değiş Tokuş Tekniği

```
void DegisTokus(Item L[],int dd,int dh,int hd,int hh) {
        int ek_alan;
        for(;dd<=dh;dd++,hd++) {
            ek_alan = L[dd];
            L[dd] = L[hd];
            L[hd] = ek_alan;
        }
}
```

2.2 Ters Çevirme Tekniği

```
void TersCevirme(Item L[],int dd ,int dh , int hh) {
        Ters(L,dd,dh);     // Ters( D )
        Ters(L,dh+1,hh);     // Ters( H )
        Ters(L,dd,hh);     // Ters (D'H')
}

void Ters(Item L[],int k,int r) {
        int ek_alan;
        while(k<r) {
            ek_alan = L[k]; L[k]=L[r]; L[r] =ek_alan;    //değiş tokuş
            k++;
            r--;
        }
}
```

2.3 Ters Permütasyon Tekniği

```
void TersPermutasyon(int L[],int dd,int dh,int hh){
        int i=0;
        int m= dh-dd+1;              // D'nin eleman sayısı
        int n= hh-dh;                // H'nin eleman sayısı
        int l=m+n;                   // DH'ın eleman sayısı
        int p= OBEB(m,n);            // |D| ve  |H|'ın ortak katlarının en büyüğü
        int hedef_indeks,kaynak_indeks;    // hedef ve kaynak indeksleri
        int ek_alan;                 // ek bellek alanı
```

```
        if(m>0 && n>0) {
            for(i=0; i<p;i++) {
                hedef_indeks=dd+i;
                kaynak_indeks = dd+i+m ;    //fi(i)
                ek_alan=L[hedef_indeks];
                do {
                    L [hedef_indeks] = L[kaynak_indeks];
                    hedef_indeks = kaynak_indeks;
                    kaynak_indeks = dd + ((kaynak_indeks + m-dd) % l);
                } while(kaynak_indeks != dd+i);
                L[hedef_indeks] = ek_alan;
            }
}}
int OBEB(int x,int y) {// x,y tamsayılarının ortak katlarının en büyüğü
        if(y==0) return x;
        else return OBEB(y,x%y);
}
```

Ek 3 İkili Arama Teknikleri Kodları

```
int IA1(int L[],int kucuk, int buyuk, int x) {
    int i;
    while(kucuk<=buyuk) {
        i = (kucuk+buyuk)/2;
        if( L[i] < x)     kucuk = i+1;
        else   buyuk = i-1;
    }
    return(kucuk);
}
int IA2(int L[],int kucuk, int buyuk, int x) {
    int i;
    while(kucuk<=buyuk) {
        i = (kucuk+buyuk)/2;
        if ( L[i] <= x)      kucuk = i+1;
        else   buyuk = i-1;
    }
    return(kucuk);
}
```

```
int IA3(int L[],int dd, int dh, int hh) {
    int i;
    int kucuk = 0;
    int buyuk = (dh-dd+1) <= (hh - dh)  ? (dh-dd) : (hh - dh -1) ;

    while(kucuk<=buyuk) {
        i = (kucuk+buyuk)/2;
        if( L[dh+1+i] < L[dh-i])    kucuk = i+1;
        else   buyuk = i-1;
        }
    return(kucuk);
}
```

Ek 4 Basit ve Standart Birleştirme Algoritmaları kodları

4.1 Basit Birleştirme Algoritması

```
void BBA (int L[],Item v[],int dd,int dh,int hh){
    int s,t;
    int lend = dh-dd+1;                  // |D|
    int lenh = hh-dh;                  // |H|
    int hd = dh+1;
    int i,j ;
    if ( lend <= lenh ) {                  // |D| ≤ |H|
        s=dd; t=dh+1;
        for( i=0;i<lend;i++)              // D'yi v'ye taşı
        v[i] = L[i+dd];
        for(i=0;i<lend;i++) {
            while(L[t] < v[i])     L[s++] = L[t++];
            //gözcüye gerek var
            /* gözcüye'e gerek duymayan için                    *
            *while((t<= hh) & (L[t] < v[i]))      L[s++] = L[t++]; */
            L[s++] = v[i];
        }
    }
    else {                          // |H| < |D|
        s=hh; t=dh;
        for( i=0;i<lenh;i++)      // D'yi v'ye taşı                    v[i] = L[i+hd];
        for(i=lenh-1;i>=0;i--){
```

```
            while(L[t]>v[i])  L[s--]=L[t--];              //gözcüye gerek var
            /* gözcüye'e gerek duymayan için              *
            * while((t>=dd) & (L[t]>v[i]))    L[s--]=L[t--]; */
            L[s--]=v[i];
        }
    }
}// BBA Sonu
```

4.2 Standart Birleştirme Algoritması

```
void SBA(int L[],int v[],int dd , int dh , int hh) {
        int i=dd,j=dh+1,k=dd;    //
        for( ; k<hh+1; k++) {
            if(i == dh+1) { v[k] = L[j++]; continue;}
            if(j == hh+1) { v[k] = L[i++]; continue;}
            v[k] = (L[i] <= L[j] ) ? L[i++] : L[j++];
        }
        for(i=dd;i<=hh;i++)      // Başlangıç dizinine kopyala
            L[i] = v[k];
}
```

Ek 5 Ayrıştırma Tabanlı Algoritmaların Kodları

5.1 AT1 Algoritması

```
void AT1(Item L[],int dd, int dh, int hh) {
    int lend= dh - dd + 1;        // D'nin eleman sayısı
    int lenh= hh - dh;          // H'nin eleman sayısı
    int s;                  // orta elemanın indeksi
    int i;                  // Aranan elemanın indeksi
    if( lend>0  &&  lenh>0 ) {
        if(lend <= lenh) {
        s = (int)ceil((double)(dh+dd)/2);
        i = IA1(L,dh+1,hh,L[s]);
        TersPermutasyon(L,s,dh,i-1);         // D2 ↔ H1
        AT1(L,dd,s-1,(s-1)+(i-1-dh));
        AT1(L, (s+1)+(i-1-dh),i-1,hh);
        }
        else {
```

```
        s = (int) ceil((double)(hh+dh+1)/2);
        i = IA2(L,dd,dh,L[s]);
        TersPermutasyon (L,i,dh,s);
        AT1(L,dd,i-1,(s-(dh+1)+(i-1)));
        AT1(L,(s-dh)+ i,s,hh);
        }
}}// AT1 sonu
```

5.2 AT2 Algoritması

```
void AT2(Item L[],int dd,int dh,int hh) {
    int lend= dh - dd + 1;            // D'nin eleman sayısı
    int lenh= hh - dh;              // H'nin eleman sayısı
    int i;                      // Değişecek eleman sayısı
    if(lend > 0 && lenh >0 ) {     // Blokların elemanları varsa
        i = IA3(L, dd,dh,hh);
        DegisTokus(L,dh-i+1,dh,dh+1,dh+i );
        AT2(L,dd,dh-i,dh);       // alt problem 1
        AT2(L,dh+1,dh+i,hh);       // alt  problem 2
    }//endif.
} // AT2 sonu.
```

5.3 ATD1 Algoritması

```
void ATD1(Item L[],int dd, int dh, int hh) {
    int lend= dh - dd + 1;            // D'nin eleman sayısı
    int lenh= hh - dh;              // H'nin eleman sayısı
    int s;                      // Orta elemanın indeksi
    int i;                      // Aranan elemanın indeksi
    int v[KESME];                   // Kesme ek alanı
    if( lend>0 && lenh>0 && L[dh]>L[dh+1]) {
        if( lend <= KESME || lenh <= KESME)  // Kesme Durumu
            BBA(L,v,dd,dh,hh);
        else {
        if(lend <= lenh) {
        s = (int)ceil((double)(dh+dd)/2);
        if(L[s]<= L[dh+1])
            ATD1(L,s+1,dh,hh);              // H1=0
        else if (L[s]>L[hh]){
            TersCevirme(L,s,dh,hh);
```

```
            ATD1(L,dd,s-1,(s-1)+(hh-dh));  // H2=0
        }
        else {
            i = IA1(L,dh+1,hh,L[s]);
            TersCevirme(L,s,dh,i-1);       // D2 ↔ H1
            ATD1(L,dd,s-1,(s-1)+(i-1-dh));
            ATD1(L, (s+1)+(i-1-dh),i-1,hh);                    }
        }
        else {
        s = (int)ceil((double)(hh+dh+1)/2);
        if(L[s]>= L[dh])  ATD1(L,dd,dh,s-1); // D2=0
        else if (L[s]<L[dd]){
            TersCevirme(L,dd,dh,s);
            ATD1(L,s-dh+dd,s,hh);         // D1=0
        }
        else {
            i = IA2(L,dd,dh,L[s]);
            TersCevirme(L,i,dh,s);          // D2 ↔ H1
            ATD1(L,dd,i-1,(s-(dh+1)+(i-1)));
            ATD1(L,(s-dh)+ i,s,hh);
        }
}}}}
```

5.4 ATY1 Algoritması

```
void ATY1(Item L[],int dd,int dh,int hh ) {
    int hd ;
    int lend,lenh;          //|D| ve |H|
    int s;                  // orta elemanın indeksi
    int i;                  //Aranan elemanın indeksi
    int l = hh;
    int x;
    int ayristirilmis;
    int v[KESME];               // Kesme alanı.
    hh = dd - 2;
     while(hh < l) {
        dd = hh + 2; dh = dd;
        while((dh<=l) & L[dh]<=L[dh+1]) dh++;//D bloğunu
        if(dh <= l){                    // H bloğunu
```

```
    hh = dh + 1; x=L[dh];
    while((hh <= l) &  (x>=L[hh])) hh=hh++;
    hh=hh-1;hd=dh+1;
}
else hh=l;
ayristirilmis=0;            // Ayrıstırılmamıs
while(hh>dh && hd>dd && L[dh]>L[hd]) {
    lend=hd-dd;lenh=hh-dh;       // |D| ve |H|
    if( lend <= KESME || lenh <= KESME) // kesme
    {   BBA(L,v,dd,dh,hh);
        if(!ayristirilmis) hh--;
        break;}
    else  {
        ayristirilmis=1;       //Ayrıstırma işlemi yapılıyor
        if(lend <= lenh) {
            s = (int)ceil((double)(dh+dd)/2);
            if(L[s]<= L[dh+1]){
                hh= s-1; break;    //H1=0
            }
            else if (L[s]>L[hh]){     // H2=0
                TersCevirme(L,s,dh,hh);
                hh=(s-1)+(hh-dh);dh=s-1; hd=s;
            }
            else {
            i = IA1(L,dh+1,hh,L[s]);
            TersCevirme(L,s,dh,i-1);  // D2 ile H1
            hh = (s-1)+(i-1-dh);      // D1H1'in yeni sınırları
            dh = s-1;hd=s;
        }
    }
    else {
        s = (int)ceil((double)(hh+dh+1)/2);
        if(L[s]>= L[dh])  hh=s-1;          // D2=0
        else if (L[s]<L[dd]){
            TersCevirme(L,dd,dh,s);
            hh=(s-1-dh)+dd-1; break;      // D1=0
        }
        else {
            i = IA2(L,dd,dh,L[s]);
```

```
                TersCevirme(L,i,dh,s);
                hh= s-(dh+1)+(i-1);          // D1H1'in yeni sınırları
                dh=i-1; hd=dh+1;
            }
}}}}}
```

5.5 ATD2 Algoritması

```
void AT2Deg(int L[],int dd,int dh,int hh) {
    int lend= dh - dd + 1;          // D'nin eleman sayısı
    int lenh= hh - dh;              // H'nin eleman sayısı
    int len_min = lend <= lenh  ? lend : lenh ;
    int i;                          // Değişecek eleman sayısı
    int hd = dh+1;                  // H'nin ilk indisi
    int v[KESME];                   // Kesme ek_alan
    if(len_min > 0 && L[dh] > L[hd]) {
        if(  len_min <= KESME )     // kesme
        BBA(L,v,dd,dh,hh);
     else {                                 // Ayrıştırma süreci
        if(L[hd-len_min] > L[dh+len_min]){
            DegisTokus(L,hd-len_min,dh,hd,dh+len_min);
            if(lend <= lenh) AT2Deg(L,hd,dh+lend,hh);
            else AT2Deg(L,dd,dh-lenh,dh);
        }
        else {
            i = IA3(L, dd,dh,hh);
            DegisTokus(L,dh-i+1,dh,hd,dh+i );
            AT2Deg(L,dd,dh-i,dh);           // alt problem 1
            AT2Deg(L,dh+1,dh+i,hh);         // alt problem 2
        }
}}}
```

5.6 ATY2 Algoritması

```
void ATY2(Item a[],int dd,int dh,int hh) {
    int lend=dh-dd+1;
    int lenh=hh-dh;
    if(lend<=lenh) ATY2Son(a,dd,dh,hh);
    else ATY2Bas(a,dd,dh,hh);
}//son ATY2
```

```
void ATY2Son(int L[],int dd,int dh,int hh) {
    int len_min;                // |D|ve|H|'ın minimumu
    int hd,x;
    int i;                  // Değişecek eleman sayısı
    int l=hh;                   // Toplam uzunluk|
    int baslangic = dd;             // başlangıç indisi
    int v[KESME];               // Kesme ek_alan
    dd = l + 1;
    while(dd > baslangic) {     // problem bulunmakta
        hh = dd - 1; hd = hh;
        while((hd>front) & L[hd]>=L[hd-1]) hd--;   // H bloğu
        if(hd > front){                     // D bloğu
            dd = hd - 1 ; x=L[hd];
            while((dd>=front) & (x<=L[dd])) dd=dd--;
            dd=dd+1;dh=hd-1;
        }
        else dd=front;
        while(hh>dh && hd>dd && L[dh]>L[hd]) {
            len_min = (dh-dd+1) <= (hh - dh) ? (dh-dd+1) : (hh - dh);
        if(  len_min <= KESME ) {   // Kesme durumu
                BBA(L,v,dd,dh,hh);
                break;              }
            else {
                if(L[hd-len_min] > L[dh+len_min]) i=len_min;
                else   i = IA3(L, dd,dh,hh);
                DegisTokus(L,dh-i+1,dh,hd,dh+i );
                dd = hd ; dh=dh+i;hd=dh+1;
            }
        }
}}//son ATY2Son

void ATY2Bas(int L[],int dd,int dh,int hh) {
    int len_min;                // |D|ve|H|'ın minimumu
    int hd,x;
    int i;                  // Değişecek eleman sayısı
    int l=hh;               // Toplam uzunluk|
    int v[KESME];               // Kesme ek_alan
    hh = dd-1;
    while(hh < l) {
            dd = hh + 1; dh = dd;
```

```
        while((dh<=l) & L[dh]<=L[dh+1]) dh++;     // D bloğu
        if(dh <= l){                              // H bloğu
        hh = dh + 1; x=L[dh];
        while((hh<=l) & (x>=L[hh])) hh=hh++;
        hh=hh-1;hd=dh+1;
      }
      else hh=l;
      while(hh>dh && hd>dd && L[dh]>L[hd]) {
        len_min = (dh-dd+1) <= (hh - dh)  ? (dh-dd+1) : (hh - dh);
      if(  len_min <= KESME ) {   // Kesme durumu
            BBA(L,v,dd,dh,hh);
        break;}
        else {
            if(L[hd-len_min] > L[dh+len_min]) i=len_min;
            else   i = IA3(L, dd,dh,hh);
            DegisTokus(L,dh-i+1,dh,hd,dh+i );
            hh=dh;  dh=    dh-i; hd=dh+1;
        }
}}}
```

Ek 6 Paralel Algoritmaların Benzetim Kodları

6.1 ATP1 Algoritması

```
void ATP1(int L[],int dd,int dh,int hh,int d,int pno){
      int lend= dh - dd + 1;              // D'nin eleman sayısı
      int lenh= hh - dh;                  // H'nin eleman sayısı
      int s;                              // orta elemanın indeksi
      int i;                              // Aranan elemanın indeksi
      int maske2= (int) pow(2,d-1);     // Yeni problem seçimi
      while(d>0) {
        d--;
        if( lend>0 && lenh>0 && a[dh]>a[dh+1]) {
            if(lend <= lenh) {
                s = (int)ceil((double)(dh+dd)/2);
                i = IA1(L,dh+1,hh,L[s]);
                TersPermutasyon(L,s,dh,i-1);          // D2 ile H1
                if( (maske2&pno) ==0){    //baştaki parça yeni problem
                hh=(s-1)+(i-1-dh); dh= s-1; }
```

```
            else {                          //sondaki parça yeni problem
                dd=(s+1)+(i-1-dh); dh=i-1; }
            }
            else {
                s = (int) ceil((double)(hh+dh+1)/2);
                i = IA2(L,dd,dh,L[s]);
                TersPermutasyon (L,i,dh,s);
                if((maske2&pno)==0){
                    hh=(s-(dh+1)+(i-1));dh= i-1; }
                else {dd=(s-dh)+ i; dh=s;  }
            }
            mask = pow(2,d-1);
        }
        else break;
    }
    ATY2(L,dd,dh,hh);
}
```

6.2 ATP2 Algoritması

```
void ATP2(Item L[],int dd,int dh,int hh,int d,int pno){
    int lend= dh - dd + 1;          // D'nin eleman sayısı
    int lenh= hh - dh;              // H'nin eleman sayısı
    int i;                          // Değişecek eleman sayısı
    int mask= (int) pow(2,d-1);
    while(d>0) {
        d--;
        if(len_min > 0 && a[dh] > a[hd]) {
            i = IA3(L, dd,dh,hh);
            DegisTokus(L,dh-i+1,dh,dh+1,dh+i );
            if( (mask&pno) ==0){    // baştaki parça yeni problem
                hh=dh; dh= dh-i;}
            else { dd=dh+1; dh=dh+i;}   // sondaki parça yeni problem
            mask = pow(2,d-1);
        }
        else break;
    }
    ATY2(L,dd,dh,hh);
}
```

Ek 7 Seri Birleştirme Gerçekleştirimleri Test Sonuçları

7.1 Test Süreleri (μS)

Değer Kümesi Eleman Sayısı = 0,001xN için Çalışma Süreleri (μs)										
m/N	N	AT1	ATD1	ATY1	AT2	ATD2	ATY2	ATY2Bas	ATY2Son	SBA
0,1	12500	3127,94	988,39	1482,05	431,77	386,04	750,09	1790,83	748,36	378,30
	25000	7258,99	2317,20	3424,24	988,75	891,32	1688,44	4242,35	1680,92	756,55
	50000	16242,57	5265,34	7705,46	2273,86	2073,41	3740,36	9603,62	3738,16	1532,85
	100000	35627,86	11714,21	16980,54	4925,35	4528,86	8009,90	20419,28	8005,72	3093,86
	200000	79249,48	25738,77	36981,81	10836,74	10040,57	17249,73	44371,91	17255,72	6177,80
	400000	177333,81	56188,45	79604,67	23733,85	22140,46	37137,35	97869,14	37151,07	12361,18
	800000	406544,78	121229,49	171318,54	50979,60	47802,50	79102,41	211554,00	79003,91	24709,38
	1600000	926552,10	260023,90	368703,55	109473,01	103106,60	167249,44	452803,73	167564,90	49412,06
	3200000	2067980,38	557424,88	783803,61	233511,46	220938,39	353110,68	968879,27	354009,49	98132,66
	6400000	4631849,22	1182552,12	1658755,31	496418,85	470937,70	745182,53	2065348,03	746055,93	196302,92
	12800000	10315947,64	2507875,38	3511651,33	1055211,65	1003970,89	1570554,25	4393384,39	1574123,16	392555,55
0,25	12500	3213,02	1015,90	1588,78	485,59	424,69	921,90	1252,87	920,83	378,13
	25000	7430,58	2406,51	3732,97	1164,54	1046,55	2101,16	2825,44	2099,48	758,95
	50000	16550,76	5421,81	8494,97	2598,80	2364,30	4632,48	6299,32	4628,54	1538,48
	100000	36403,02	12040,24	18158,33	5860,44	5391,12	10133,20	13860,31	10134,29	3094,21
	200000	80565,65	26397,05	39158,62	12814,34	11861,45	21598,11	30092,55	21791,92	6188,26
	400000	180250,27	57301,37	84477,66	28064,19	26314,73	46522,53	65412,41	46589,73	12378,88
	800000	408763,97	123840,79	180830,62	60149,33	56303,87	98898,81	140938,59	98800,56	24747,76
	1600000	925053,77	265174,66	386652,03	130277,38	122606,31	211021,74	301465,06	211231,51	49504,23
	3200000	2074211,50	565912,91	817145,12	276119,06	260689,95	445215,94	643919,41	444195,57	98295,24
	6400000	4635447,33	1202868,26	1728659,10	592867,64	562263,08	943633,65	1364301,41	944641,37	196647,97
	12800000	10233295,58	2549258,83	3649796,87	1248851,58	1186078,40	1975454,24	2892657,17	1977253,71	393839,91
0,5	12500	3135,64	986,29	1448,54	537,15	447,30	1188,01	1186,84	1174,84	380,90
	25000	7326,93	2370,18	3367,38	1297,61	1123,93	2652,63	2650,36	2605,74	760,32
	50000	16255,81	5333,61	7508,63	2909,16	2558,66	5759,09	5751,91	5700,48	1537,71
	100000	35636,00	11830,29	16663,77	6428,18	5748,40	12424,04	12391,93	12393,42	3099,02
	200000	78712,10	25964,58	35872,30	14108,90	12754,04	26650,08	26616,82	26836,41	6200,30
	400000	176282,39	56589,02	77622,74	30688,91	27959,46	57060,37	57019,81	57598,39	12392,26
	800000	398118,45	121957,40	166973,54	66183,64	60759,62	121847,23	121499,40	123363,13	24776,27
	1600000	905786,85	261245,71	356897,39	141910,93	131172,09	257350,98	257163,68	262544,13	49557,46
	3200000	2016171,45	558307,54	760639,28	303234,01	281708,15	545653,31	544305,86	555744,00	98498,48
	6400000	4520793,45	1187903,24	1614833,20	647069,74	603646,58	1150302,32	1151950,93	1178162,63	196946,49
	12800000	10055766,85	2519224,01	3423495,62	1368622,51	1284125,23	2424673,49	2422429,46	2481072,06	394402,35
0,75	12500	3419,90	1030,57	1407,75	485,94	423,49	934,13	933,21	1263,70	378,67
	25000	7829,43	2416,37	3261,61	1167,29	1047,55	2128,37	2126,82	2784,91	762,26
	50000	17437,19	5472,98	7340,81	2601,55	2364,08	4654,24	4644,84	6264,44	1547,72
	100000	38299,76	12124,53	16186,19	5882,54	5392,33	10174,00	10154,75	14064,31	3109,78
	200000	83084,63	26571,40	35291,88	12885,39	11925,14	21715,48	21691,04	30782,87	6218,84
	400000	187189,98	57638,62	76452,28	28196,79	26244,74	46625,65	46768,13	67473,39	12437,37
	800000	421715,22	124293,77	164677,07	60432,24	56525,02	98785,64	98564,17	145814,44	24866,79
	1600000	956711,54	266459,71	351993,87	130835,02	123008,01	212058,40	210357,94	313555,04	49761,19
	3200000	2135135,84	568858,97	750173,78	277753,83	261418,11	441179,07	440396,67	670496,45	98842,96
	6400000	4741755,47	1208321,87	1592345,18	598951,75	565817,07	936512,26	934740,68	1418215,29	197855,92
	12800000	10449582,76	2559807,08	3370918,15	1253617,25	1189748,60	1953740,84	1952378,72	3010868,23	396747,34
0,9	12500	3396,94	1016,60	1275,59	435,06	387,23	762,32	761,26	1756,24	378,90
	25000	7803,53	2345,17	2996,15	995,69	891,38	1689,70	1689,34	4136,23	762,06
	50000	17360,98	5326,23	6802,89	2290,08	2080,36	3790,22	3785,27	9681,47	1545,28
	100000	37993,01	11832,54	15092,48	4967,82	4549,85	8202,00	8198,47	20925,18	3117,23
	200000	83336,54	25990,50	33129,28	10977,22	10150,20	17387,81	17376,65	46642,55	6235,47
	400000	187189,45	56529,59	72070,17	24096,24	22434,27	37345,13	37331,16	103200,26	12468,35
	800000	425155,73	122205,84	155583,59	51805,33	48340,38	79156,41	79181,47	224830,35	24940,76
	1600000	959410,40	262004,97	335769,94	110495,95	103829,46	167735,46	167794,94	484846,34	49872,11
	3200000	2153131,50	559787,52	716014,92	235886,35	222619,51	353854,43	353311,42	1037260,93	99248,71
	6400000	4799350,44	1192566,36	1520785,06	501655,10	474631,41	743257,42	743119,13	2203881,64	198758,31
	12800000	10628690,94	2524107,19	3222297,92	1068010,39	1013125,22	1563038,03	1565798,97	4689019,65	398502,84

Değer Kümesi Eleman Sayısı = 0,01xN için Çalışma Süreleri (μs)										
m/N	N	AT1	ATD1	ATY1	AT2	ATD2	ATY2	ATY2Bas	ATY2Son	SBA
0,1	12500	3685,37	1173,92	1602,68	860,03	614,06	883,46	2366,94	881,85	384,45
	25000	8105,63	2624,71	3598,80	1837,02	1327,13	1896,86	5221,19	1894,74	765,06
	50000	17680,39	5807,67	7976,24	3918,66	2889,51	4067,41	11517,95	4064,32	1548,71
	100000	38371,92	12725,93	17470,72	8298,32	6288,66	8785,68	24913,15	8780,21	3124,65
	200000	84423,54	27654,71	37925,38	17416,07	13470,39	18767,97	53870,13	18773,05	6242,12
	400000	187676,86	59750,40	81927,02	36633,26	28672,66	39797,59	115845,45	39790,07	12482,36
	800000	424969,86	128624,15	175556,77	76975,22	60987,92	84028,53	243927,33	84095,53	25136,40
	1600000	966106,49	274525,14	372748,39	161090,95	129569,49	177401,41	516416,29	177577,94	49905,65
	3200000	2149182,87	584320,38	792895,42	336420,77	273923,02	373741,76	1092741,31	373938,83	99281,41
	6400000	4791590,77	1239784,41	1680623,98	702059,83	576166,83	783972,23	2310406,59	785233,18	198412,15
	12800000	10607263,28	2624387,94	3555913,03	1465339,54	1210596,58	1641777,66	4880761,48	1645833,17	397341,26
0,25	12500	4014,16	1319,19	1843,17	991,49	789,05	1140,88	1719,87	1157,57	383,83
	25000	8783,23	2911,36	4069,76	2111,74	1706,80	2462,32	3768,21	2460,41	786,98
	50000	18991,99	6377,40	8905,64	4514,10	3705,14	5312,97	8224,67	5310,91	1553,91
	100000	41149,11	13862,21	19297,46	9563,59	7949,48	11387,92	17533,01	11384,55	3126,64
	200000	89299,12	29955,58	41537,17	20256,09	17046,49	24306,18	37723,81	24515,50	6249,02
	400000	197574,10	64337,24	89041,30	42603,29	36220,86	51680,78	80296,41	51508,29	12495,90
	800000	442770,10	137547,38	189158,25	89707,80	77000,45	108948,68	169837,75	109033,88	24981,12
	1600000	991321,13	292784,40	400904,06	187738,64	162543,57	229523,79	356852,11	229710,30	49979,24
	3200000	2215496,42	620887,87	848190,33	393136,84	342971,45	483317,58	752275,95	483427,86	99417,17
	6400000	4908316,69	1314608,43	1790441,23	821228,99	722778,77	1013846,70	1586478,36	1014569,40	200503,04
	12800000	10785348,72	2769914,68	3774904,33	1713986,61	1514888,58	2124949,75	3335115,31	2125021,59	398177,00
0,5	12500	4096,54	1396,18	1942,38	1067,67	829,20	1341,94	1339,08	1329,76	386,66
	25000	8924,67	3065,04	4272,74	2280,44	1807,23	2931,73	2927,71	2904,62	771,53
	50000	19309,97	6684,55	9252,17	4862,06	3916,40	6358,02	6351,27	6318,43	1555,15
	100000	41615,96	14507,90	19974,61	10319,67	8449,02	13722,15	13689,94	13608,92	3131,68
	200000	90331,14	31227,96	42830,30	21857,27	18117,57	29228,47	29181,59	29180,44	6262,14
	400000	198946,18	66887,21	91459,21	46153,03	38635,81	62353,85	62045,83	62408,02	12516,85
	800000	443392,14	142590,77	193861,96	96904,27	82220,08	131595,04	131210,03	132021,50	25017,65
	1600000	992240,08	302853,52	412267,28	203239,97	173625,55	277503,62	277203,08	279488,76	50055,53
	3200000	2197012,14	640850,85	866617,59	425082,27	366106,78	585400,59	584040,26	590903,92	99480,24
	6400000	4913774,25	1371342,36	1830977,86	891437,34	771312,21	1231190,90	1230627,41	1244896,25	199006,28
	12800000	10791660,29	2858822,16	3842272,42	1856295,48	1617688,03	2583890,46	2585385,69	2605375,75	398946,14
0,75	12500	4203,34	1330,42	1751,25	995,15	789,13	1153,09	1150,70	1699,70	387,60
	25000	9145,82	2937,09	3866,36	2117,44	1713,16	2504,45	2483,62	3722,04	775,49
	50000	19752,11	6449,58	8466,48	4531,18	3717,25	5386,43	5562,97	8225,37	1565,30
	100000	42561,58	14029,37	18368,65	9589,68	7973,99	11484,75	11473,07	17686,64	3145,20
	200000	91706,91	30228,91	39727,38	20355,83	17130,67	24718,36	24500,22	38229,74	6285,11
	400000	204255,68	64878,03	84815,48	42810,31	36376,03	51902,05	51836,11	81826,56	12563,59
	800000	456014,84	138534,28	180715,34	90124,94	77377,51	109727,32	109635,72	174412,50	25108,95
	1600000	1019850,69	294465,43	383725,91	188674,62	163192,04	230867,60	230834,53	369939,65	50197,18
	3200000	2260536,58	624038,07	813005,22	394881,74	344275,93	488055,01	485792,50	780188,98	99949,77
	6400000	4982536,30	1320726,64	1717772,91	825299,04	725270,80	1019033,17	1018525,44	1645872,69	199962,90
	12800000	10983920,28	2782543,97	3620522,36	1723160,66	1516915,31	2133814,55	2137870,07	3456421,00	401031,22
0,9	12500	3947,48	1189,18	1522,80	871,82	619,02	895,64	893,71	2335,83	388,37
	25000	8651,83	2655,59	3397,21	1866,41	1339,64	1925,74	1923,93	5133,42	776,45
	50000	18747,71	5880,87	7515,90	3987,07	2917,21	4128,44	4126,45	11220,99	1566,68
	100000	40794,62	12884,66	16442,62	8445,57	6365,52	8899,20	8899,57	25278,50	3152,72
	200000	88357,68	27984,42	35712,51	17760,71	13650,22	19037,30	19044,21	55577,70	6300,82
	400000	196534,03	60391,48	77098,93	37405,51	29106,30	40491,39	40305,58	120282,28	12593,67
	800000	443595,66	129643,88	165301,69	78483,52	61745,83	84996,69	85008,62	258189,04	25182,39
	1600000	995839,10	276628,87	352813,76	163783,86	130798,06	179345,02	179471,04	554148,84	50393,03
	3200000	2237793,95	588372,70	750925,64	341810,10	276212,35	377565,15	377748,80	1174871,79	100216,81
	6400000	4966695,63	1247508,46	1594375,44	714212,50	581194,85	791740,25	792726,58	2475810,87	200739,56
	12800000	10910776,36	2639135,33	3369192,38	1488044,15	1220882,64	1658910,20	1659260,91	5221818,38	403305,32

Değer Kümesi Eleman Sayısı = 0,1xN için Çalışma Süreleri (μs)										
m/N	N	AT1	ATD1	ATY1	AT2	ATD2	ATY2	ATY2Bas	ATY2Son	SBA
0,1	12500	4049,05	1222,67	1766,55	1405,10	740,96	965,43	2500,06	963,95	405,67
	25000	8815,18	2716,56	3886,24	2917,41	1559,79	2036,75	5472,83	2035,33	816,08
	50000	19074,18	5982,03	8544,32	6039,03	3345,63	4334,56	12041,50	4334,22	1632,26
	100000	41042,33	13064,93	18592,46	12549,76	7232,01	9331,97	26156,32	9344,91	3298,14
	200000	88557,36	28340,66	39972,33	25889,21	15381,74	19828,98	55608,52	19861,35	6586,51
	400000	197468,01	61270,21	85563,96	53292,70	32341,03	41818,86	116724,26	41830,38	13174,56
	800000	445555,27	131050,76	182706,71	109705,45	68150,26	88381,96	248535,70	88161,87	26328,15
	1600000	984395,00	279810,32	388382,34	225424,30	143995,28	185636,24	528199,58	185850,36	52608,98
	3200000	2232184,86	595034,99	824932,22	465012,43	302878,87	392112,33	1119456,00	390588,29	104478,69
	6400000	4900328,86	1263061,93	1745666,59	958975,46	633901,46	816535,00	2367342,18	817048,25	209046,76
	12800000	10779500,14	2665054,71	3690952,19	1976108,73	1325808,09	1707148,23	4993711,54	1709256,14	418329,36
0,25	12500	4863,88	1492,75	2045,76	1789,58	1001,05	1330,32	1914,59	1328,37	423,67
	25000	10460,43	3252,94	4472,43	3709,41	2154,39	2847,98	4176,28	2846,97	857,44
	50000	22337,05	7058,45	9705,77	7693,12	4546,81	6035,82	9040,37	6041,07	1716,85
	100000	47631,69	15214,37	20881,54	15885,99	9655,63	12849,08	19275,69	12854,99	3451,67
	200000	102735,37	32645,19	44495,58	32910,33	20375,86	27174,41	40497,61	27172,55	6903,22
	400000	223416,45	69699,11	94501,61	67442,03	42921,94	57465,10	85446,45	57318,49	13807,70
	800000	494115,03	148267,14	200764,08	139037,28	90321,38	120235,09	180602,44	120429,77	27616,52
	1600000	1096306,15	314132,42	424582,11	286459,32	188846,87	252862,54	380417,58	252300,87	55319,91
	3200000	2416440,84	663795,49	896381,49	586218,16	395537,44	528082,92	800450,50	527698,04	110097,42
	6400000	5320453,40	1400259,25	1887865,29	1208449,27	826787,03	1104757,37	1682631,94	1103720,54	220396,76
	12800000	11603801,40	2940023,01	3972169,33	2488877,60	1726090,82	2305487,58	3528432,74	2305571,18	442435,41
0,5	12500	5310,55	1783,35	2326,99	2040,17	1220,37	1738,52	1734,79	1731,50	432,64
	25000	11339,48	3834,53	5018,73	4212,50	2585,46	3717,67	3712,63	3703,68	864,43
	50000	24089,12	8401,49	10772,81	8698,83	5468,71	7921,13	7909,49	7896,52	1742,86
	100000	51192,96	17348,48	22967,59	18040,85	11547,76	16757,51	16721,48	16759,22	3509,32
	200000	109442,24	37302,42	48599,67	37124,09	24284,53	35179,66	35111,81	35378,21	7022,78
	400000	237152,72	79015,84	102842,75	76597,85	50921,53	73992,14	73868,00	74847,82	14049,39
	800000	520081,04	166823,14	217230,47	157314,44	106557,54	155416,70	155208,34	156522,98	28126,92
	1600000	1145330,91	351280,17	457214,17	323004,82	222520,84	325490,73	325115,69	327899,86	56351,24
	3200000	2509010,28	737480,11	960445,16	664324,90	465989,07	680853,71	680604,01	687594,27	113326,23
	6400000	5485202,03	1547198,97	2016522,02	1364172,66	965938,50	1422209,12	1421094,31	1432706,60	225910,39
	12800000	12024379,98	3233126,79	4223293,87	2804690,68	2008912,25	2972325,80	2966013,15	2989838,24	455508,35
0,75	12500	4970,41	1506,42	1929,76	1847,39	1002,92	1341,18	1338,23	1912,93	428,85
	25000	10669,06	3298,96	4213,86	3822,71	2150,34	2869,06	2865,76	4164,73	857,40
	50000	22729,83	7115,91	9143,99	7910,71	4569,21	6122,25	6117,95	8912,20	1729,45
	100000	48494,22	15336,94	19701,46	16375,40	9699,58	13169,54	12981,90	19350,78	3478,80
	200000	103741,59	32868,04	42138,78	33808,05	20503,44	27393,40	27379,32	41444,52	6959,35
	400000	227372,60	70138,50	89880,67	69729,59	43144,65	57628,95	57775,24	88172,81	13917,09
	800000	503288,67	149520,71	191107,79	143723,81	90659,26	121106,41	121252,45	186653,88	27908,54
	1600000	1117112,83	315825,79	404544,65	294815,82	190001,21	253935,19	253955,05	394444,69	55879,07
	3200000	2449446,95	668469,70	854394,74	605951,10	399044,67	531875,69	531713,11	831312,61	113874,70
	6400000	5384616,36	1405860,12	1801344,62	1249414,09	829810,96	1113945,97	1111978,21	1743633,41	224285,01
	12800000	11757825,29	2952587,80	3789029,89	2569215,25	1731757,54	2320894,35	2322750,73	3654862,41	447837,15
0,9	12500	4265,95	1238,66	1576,74	1497,51	745,45	971,37	969,36	2485,25	416,33
	25000	9245,00	2746,48	3497,32	3096,61	1568,90	2049,98	2065,26	5436,86	832,33
	50000	19914,83	6035,44	7686,05	6413,74	3361,80	4387,03	4383,13	11819,92	1673,48
	100000	43117,53	13178,84	16774,45	13318,23	7277,65	9444,02	9440,83	26476,74	3365,50
	200000	92716,96	28567,78	36341,09	27477,30	15488,83	20255,67	20080,73	57701,79	6701,72
	400000	205809,68	61563,53	78328,14	56661,89	32660,61	42550,71	42358,90	124109,08	13400,76
	800000	461615,65	132418,90	168146,65	116593,20	68839,18	89035,26	89039,00	265469,78	26803,50
	1600000	1033130,83	281629,21	358313,14	239417,64	145190,44	187969,63	187961,77	566728,39	53638,63
	3200000	2308666,26	598750,97	762039,51	493668,93	304931,37	395163,52	395112,71	1205081,68	106624,24
	6400000	5104104,90	1270497,73	1614625,03	1018547,40	638212,47	826580,26	828289,68	2549517,77	213185,41
	12800000	11206269,47	2682150,14	3413441,59	2094272,50	1334939,10	1732209,00	1728854,32	5362437,31	426547,91

Değer Kümesi Eleman Sayısı = N için Çalışma Süreleri (μs)										
m/N	N	AT1	ATD1	ATY1	AT2	ATD2	ATY2	ATY2Bas	ATY2Son	SBA
0,1	12500	4101,81	1228,14	1785,57	1477,99	748,11	918,99	2528,52	917,16	416,14
	25000	8917,85	2728,05	3948,78	3066,02	1574,98	1950,26	5469,79	1949,02	808,54
	50000	19273,91	6003,35	8668,59	6360,28	3376,93	4168,86	12140,67	4174,46	1633,91
	100000	41502,58	13111,13	18720,95	13143,11	7288,36	8997,49	25460,01	9019,93	3299,28
	200000	90547,11	28420,22	40100,39	27089,40	15489,16	19232,45	54678,12	19227,88	6579,12
	400000	199470,10	61232,45	85867,30	55638,47	32629,40	40632,91	117205,65	40836,21	13174,53
	800000	450689,32	131339,22	183670,65	114602,96	68655,63	85800,43	248909,62	86436,96	26334,36
	1600000	1006485,46	280342,28	390734,44	234312,99	144998,52	181227,66	529645,76	181376,84	52638,86
	3200000	2250247,91	596295,22	829734,71	482994,67	305114,70	381784,47	1123725,97	383701,19	104547,68
	6400000	4989002,25	1263547,54	1757754,78	997285,27	638162,81	799633,54	2377024,04	801985,40	209022,03
	12800000	10963490,48	2673743,09	3710481,62	2050474,66	1336219,70	1675189,21	5013422,50	1674977,85	417902,26
0,25	12500	5068,68	1525,85	2116,63	2033,98	1040,51	1305,50	1976,76	1304,81	453,03
	25000	10862,14	3320,14	4603,91	4195,57	2215,95	2801,24	4287,96	2801,11	902,77
	50000	23130,70	7190,46	9966,60	8665,30	4704,16	5952,48	9307,39	5963,84	1824,99
	100000	49198,04	15475,61	21495,42	17819,72	9971,55	12690,95	19522,64	12701,62	3665,88
	200000	105687,46	33172,43	45325,51	36567,54	21027,16	26926,31	41495,11	26936,38	7314,51
	400000	229997,98	70762,89	96532,38	75031,84	44240,78	57040,75	87507,77	56849,00	14638,08
	800000	508501,96	150370,52	204609,51	154062,01	93007,94	119786,00	184634,55	119483,07	29289,24
	1600000	1122059,47	318439,65	432504,69	315683,97	194380,76	251127,32	388409,62	252934,41	58564,62
	3200000	2471519,26	672377,45	912557,33	649155,76	406296,98	526160,98	817356,84	525775,80	118251,74
	6400000	5437183,44	1415724,51	1922393,59	1330760,00	849891,24	1099367,63	1714769,54	1099998,76	232803,01
	12800000	11819594,07	2977504,56	4033897,28	2736734,34	1767053,86	2297138,79	3599253,60	2295379,56	465696,52
0,5	12500	5643,51	1842,96	2396,93	2457,44	1290,26	1810,35	1808,45	1793,73	500,02
	25000	11972,53	3954,87	5155,95	5047,10	2725,04	3857,97	3855,40	3831,84	1000,33
	50000	25324,69	8455,76	11048,71	10372,54	5746,44	8204,45	8371,07	8154,75	2009,32
	100000	53849,82	18017,38	23461,00	21310,75	12096,77	17290,90	17262,16	17292,38	4051,00
	200000	114323,70	38247,11	49617,81	43736,97	25400,17	36483,27	36262,73	36542,56	8102,88
	400000	247162,41	80908,10	104944,44	89911,62	53197,56	76297,82	76394,29	76957,85	16199,03
	800000	540549,45	170676,13	221579,42	183599,71	111145,75	160373,81	159930,06	161595,76	32402,56
	1600000	1185742,18	358993,53	466535,62	375896,81	231557,17	335099,86	335624,86	337868,81	64739,65
	3200000	2587975,66	753345,15	980357,84	768802,74	482849,62	699897,26	699588,36	706464,89	128754,64
	6400000	5661104,72	1577297,96	2052314,59	1575301,51	1005251,49	1459890,88	1461904,67	1473835,42	257671,68
	12800000	12355575,43	3298542,25	4291315,70	3229202,82	2083412,36	3045470,15	3045146,87	3071771,46	514240,46
0,75	12500	5157,87	1535,18	1956,32	2133,47	1040,00	1328,08	1326,28	1978,90	456,18
	25000	11004,96	3339,82	4268,24	4395,67	2224,87	2836,77	2835,63	4285,77	911,10
	50000	23412,27	7218,65	9243,36	9064,52	4718,19	6055,83	6054,54	9242,97	1838,47
	100000	49837,61	15548,05	19872,69	18668,91	9992,97	12839,70	12841,21	19804,18	3694,63
	200000	106198,37	33493,64	42512,15	38392,58	21091,37	27152,78	27153,27	42478,38	7384,92
	400000	232399,88	71044,26	90690,96	78814,64	44353,75	57245,78	57243,41	90416,10	14755,49
	800000	514204,22	151093,88	192771,08	161848,26	93134,71	120564,69	120373,52	191916,20	29681,81
	1600000	1151998,00	325260,88	415341,18	337291,44	198620,98	259553,79	256422,41	408749,80	58981,22
	3200000	2493175,19	674604,97	862269,83	679694,46	409136,47	529573,36	530520,10	855208,74	117453,82
	6400000	5477689,24	1419783,78	1819615,70	1394584,78	850052,06	1106644,52	1107028,53	1791296,82	234831,30
	12800000	11912941,86	2985177,60	3818196,05	2861404,23	1772209,50	2311486,53	2312891,05	3752879,06	469695,63
0,9	12500	4302,19	1241,94	1582,07	1583,91	750,07	930,88	928,71	2519,58	416,93
	25000	9313,26	2752,60	3508,11	3273,72	1578,85	1973,24	1970,61	5510,01	833,94
	50000	20068,53	6045,65	7708,19	6773,21	3381,28	4237,39	4231,80	12046,36	1676,00
	100000	43196,99	13194,62	16815,30	14010,55	7316,15	9133,12	9125,10	26441,59	3353,04
	200000	93192,76	28595,64	36606,15	28874,45	15574,96	19460,46	19457,51	57466,41	6711,18
	400000	206639,69	61804,78	78495,36	59512,04	32838,69	41102,55	41107,33	124342,35	13420,80
	800000	463869,03	132173,36	168291,42	122436,09	69200,93	86778,18	86786,76	266888,50	26809,44
	1600000	1039579,71	281928,24	359114,44	250850,75	145976,25	183453,19	183466,16	571567,34	53559,20
	3200000	2312357,87	599380,93	765296,46	517308,18	308439,23	386120,60	386263,61	1219157,99	106266,44
	6400000	5109640,35	1269875,07	1618084,29	1062859,46	642033,43	808665,35	810469,88	2587778,04	212579,24
	12800000	11224157,06	2686111,34	3417929,94	2188157,10	1341611,97	1692441,97	1695714,72	5457747,34	425469,53

Değer Kümesi Eleman Sayısı = 10xN için Çalışma Süreleri (μs)										
m/N	N	AT1	ATD1	ATY1	AT2	ATD2	ATY2	ATY2Bas	ATY2Son	SBA
0,1	12500	4117,02	1227,78	1788,32	1478,50	748,25	910,80	2529,49	909,12	415,14
	25000	8906,73	2727,76	3953,83	3067,33	1575,16	1918,54	5473,18	1934,94	806,59
	50000	19252,32	6020,69	8682,25	6363,32	3377,42	4137,77	12148,66	4142,89	1629,97
	100000	41479,53	13113,12	18746,50	13152,40	7286,75	8927,37	25649,39	8944,55	3291,33
	200000	90342,52	28417,84	40143,04	27082,45	15487,13	19437,59	54507,81	19076,79	6563,26
	400000	199520,99	61229,66	85957,11	55646,64	32629,44	40539,42	117261,90	40560,56	13142,77
	800000	450799,00	131338,99	184239,99	114450,62	68646,07	85231,47	248847,40	86060,42	26270,87
	1600000	1007904,97	280640,72	391202,38	233634,86	145124,55	180056,34	529958,03	180251,80	52511,95
	3200000	2248591,27	596143,89	830719,82	483568,61	304749,87	379682,50	1123970,24	357141,31	104295,62
	6400000	4941278,67	1263535,65	1758758,43	994268,90	638005,74	794815,67	2377407,86	797618,65	208518,08
	12800000	10897774,99	2673369,68	3712619,36	2054706,28	1335681,35	1665350,39	4988449,51	1665476,76	416894,70
0,25	12500	5068,03	1526,17	2128,66	2033,15	1041,02	1317,18	1971,70	1297,81	458,58
	25000	10878,93	3339,10	4593,43	4190,01	2216,45	2823,82	4277,63	2822,98	893,97
	50000	23085,60	7191,79	9945,02	8659,44	4705,32	6000,31	9289,08	5991,86	1822,59
	100000	49196,89	15477,15	21450,09	18025,17	9973,44	12793,07	19309,44	12802,99	3657,71
	200000	105732,73	33180,72	45241,44	36510,73	21032,39	26950,80	41424,97	27138,97	7305,09
	400000	229608,70	70609,28	96344,62	74924,70	44067,77	57443,56	87362,45	57251,47	14609,22
	800000	509586,78	150417,43	204069,01	154025,57	93014,42	120594,88	184510,62	119749,46	29045,66
	1600000	1112347,13	318539,38	438180,96	315248,76	194427,19	252920,61	387692,27	254644,13	60070,28
	3200000	2471950,49	672393,01	911274,85	651341,58	406730,71	528706,24	816110,02	528412,26	118209,16
	6400000	5463255,19	1415954,02	1919856,34	1325691,78	849256,98	1105351,24	1712120,61	1108005,35	233930,30
	12800000	11802273,93	2975742,54	4014265,60	2731871,61	1766986,95	2301533,72	3604154,57	2301637,65	464903,96
0,5	12500	5641,98	1843,88	2402,46	2454,93	1291,01	1795,91	1726,82	1722,97	499,30
	25000	11962,69	3976,05	5168,58	5041,96	2727,58	3869,43	3672,08	3677,86	998,52
	50000	25282,45	8462,07	11075,52	10383,96	5733,24	8230,10	8033,49	7821,37	2006,70
	100000	53839,83	18027,01	23517,54	21277,66	12101,70	17341,40	16780,84	16632,15	4045,89
	200000	114286,43	38084,67	49722,21	43684,75	25409,87	36576,28	34940,78	35169,57	8093,61
	400000	247031,49	80955,63	105207,92	89595,80	53221,57	76470,05	73737,96	74353,55	16177,11
	800000	540122,80	170942,94	221502,70	183349,42	111370,39	160728,64	154616,51	155771,67	32358,00
	1600000	1198600,81	359257,11	468033,36	375223,25	231198,72	335684,71	324962,13	326504,74	64708,38
	3200000	2593927,51	753963,74	982649,18	769205,52	483230,89	701030,11	678299,57	657873,85	128402,96
	6400000	5620225,36	1578357,98	2056383,32	1576687,64	1005960,63	1461702,11	1420533,32	1427659,68	256739,32
	12800000	12363693,46	3300859,41	4324558,08	3223465,05	2083333,47	3051296,80	2962451,05	3004390,80	514233,43
0,75	12500	5153,77	1536,44	1962,25	2130,22	1041,52	1336,42	1333,83	1974,01	455,64
	25000	10992,75	3340,75	4260,41	4391,65	2222,43	2855,89	2852,93	4278,31	909,57
	50000	23412,05	7220,73	9262,42	9075,55	4722,85	6099,75	6079,01	9208,93	1836,39
	100000	49816,44	15556,44	19914,41	18655,81	9998,19	12919,51	12919,41	19768,88	3689,31
	200000	106144,23	33500,84	42412,43	38356,93	21099,55	27308,75	27308,98	42414,14	7375,31
	400000	232610,24	71240,90	90484,40	78694,23	44361,26	57380,02	57544,73	90308,12	14929,98
	800000	513457,86	151100,16	193045,56	161602,31	93323,16	120991,36	121135,70	192060,22	29640,68
	1600000	1134319,05	325342,03	416642,38	336438,83	198917,48	260375,09	257477,03	408038,18	58904,12
	3200000	2462977,68	657877,33	863367,06	678489,85	408788,29	531681,54	533067,48	855099,28	117203,10
	6400000	5475958,54	1420738,10	1822914,98	1422045,99	849130,32	1111963,10	1112114,04	1788812,72	234813,78
	12800000	12091050,71	2961849,17	3824265,63	2855934,11	1772458,40	2321522,49	2323168,35	3753447,64	469400,06
0,9	12500	4292,73	1240,64	1579,66	1583,38	749,69	936,34	932,82	2515,02	416,58
	25000	9316,84	2752,90	3505,70	3272,94	1582,20	1987,34	1982,03	5509,10	854,20
	50000	20041,09	6042,53	7697,46	6770,60	3385,17	4266,76	4261,22	12039,31	1673,96
	100000	43137,48	13188,89	16797,55	13999,98	7320,41	9189,49	9182,77	26625,35	3350,85
	200000	93266,77	28580,51	36565,08	28861,74	15580,37	19576,44	19572,37	57660,12	6915,66
	400000	207235,31	61808,97	78437,05	59512,34	32856,79	41497,99	41328,53	124294,27	13414,02
	800000	463884,09	132155,56	167787,35	122385,47	69206,28	87215,52	87399,60	267183,65	26794,41
	1600000	1032673,14	281974,61	358742,24	250829,84	145819,50	184398,61	184371,49	571711,36	53528,94
	3200000	2319744,27	599478,85	765212,01	543314,89	308827,49	388099,57	388053,21	1219071,48	106281,55
	6400000	5078603,41	1269986,05	1617964,38	1062211,49	641942,89	812502,04	814649,87	2585773,14	212493,75
	12800000	11187070,70	2686961,64	3415710,63	2186931,19	1366585,42	1699991,05	1703839,64	5448744,46	424767,68

7.2 Değer Kümesi Eleman Sayısı = N iken SBA Sürelerine Göre Normalleştirilmiş Süreler

m+n	m/N = 0,1							
N	AT1	ATD1	ATY1	AT2	ATD2	ATY2	ATY2Bas	ATY2Son
12500	9,86	2,95	4,29	3,55	1,80	2,21	6,08	2,20
25000	11,03	3,37	4,88	3,79	1,95	2,41	6,77	2,41
50000	11,80	3,67	5,31	3,89	2,07	2,55	7,43	2,55
100000	12,58	3,97	5,67	3,98	2,21	2,73	7,72	2,73
200000	13,76	4,32	6,10	4,12	2,35	2,92	8,31	2,92
400000	15,14	4,65	6,52	4,22	2,48	3,08	8,90	3,10
800000	17,11	4,99	6,97	4,35	2,61	3,26	9,45	3,28
1600000	19,12	5,33	7,42	4,45	2,75	3,44	10,06	3,45
3200000	21,52	5,70	7,94	4,62	2,92	3,65	10,75	3,67
6400000	23,87	6,05	8,41	4,77	3,05	3,83	11,37	3,84
12800000	26,23	6,40	8,88	4,91	3,20	4,01	12,00	4,01
m+n	m/N = 0,25							
12500	11,19	3,37	4,67	4,49	2,30	2,88	4,36	2,88
25000	12,03	3,68	5,10	4,65	2,45	3,10	4,75	3,10
50000	12,67	3,94	5,46	4,75	2,58	3,26	5,10	3,27
100000	13,42	4,22	5,86	4,86	2,72	3,46	5,33	3,46
200000	14,45	4,54	6,20	5,00	2,87	3,68	5,67	3,68
400000	15,71	4,83	6,59	5,13	3,02	3,90	5,98	3,88
800000	17,36	5,13	6,99	5,26	3,18	4,09	6,30	4,08
1600000	19,16	5,44	7,39	5,39	3,32	4,29	6,63	4,32
3200000	20,90	5,69	7,72	5,49	3,44	4,45	6,91	4,45
6400000	23,36	6,08	8,26	5,72	3,65	4,72	7,37	4,73
12800000	25,38	6,39	8,66	5,88	3,79	4,93	7,73	4,93
m+n	m/N = 0,5							
12500	11,29	3,69	4,79	4,91	2,58	3,62	3,62	3,59
25000	11,97	3,95	5,15	5,05	2,72	3,86	3,85	3,83
50000	12,60	4,21	5,50	5,16	2,86	4,08	4,17	4,06
100000	13,29	4,45	5,79	5,26	2,99	4,27	4,26	4,27
200000	14,11	4,72	6,12	5,40	3,13	4,50	4,48	4,51
400000	15,26	4,99	6,48	5,55	3,28	4,71	4,72	4,75
800000	16,68	5,27	6,84	5,67	3,43	4,95	4,94	4,99
1600000	18,32	5,55	7,21	5,81	3,58	5,18	5,18	5,22
3200000	20,10	5,85	7,61	5,97	3,75	5,44	5,43	5,49
6400000	21,97	6,12	7,96	6,11	3,90	5,67	5,67	5,72
12800000	24,03	6,41	8,34	6,28	4,05	5,92	5,92	5,97
m+n	m/N = 0,75							
12500	11,31	3,37	4,29	4,68	2,28	2,91	2,91	4,34
25000	12,08	3,67	4,68	4,82	2,44	3,11	3,11	4,70
50000	12,73	3,93	5,03	4,93	2,57	3,29	3,29	5,03
100000	13,49	4,21	5,38	5,05	2,70	3,48	3,48	5,36
200000	14,38	4,54	5,76	5,20	2,86	3,68	3,68	5,75
400000	15,75	4,81	6,15	5,34	3,01	3,88	3,88	6,13
800000	17,32	5,09	6,49	5,45	3,14	4,06	4,06	6,47
1600000	19,53	5,51	7,04	5,72	3,37	4,40	4,35	6,93
3200000	21,23	5,74	7,34	5,79	3,48	4,51	4,52	7,28
6400000	23,33	6,05	7,75	5,94	3,62	4,71	4,71	7,63
12800000	25,36	6,36	8,13	6,09	3,77	4,92	4,92	7,99
m+n	m/N = 0,9							
12500	10,32	2,98	3,79	3,80	1,80	2,23	2,23	6,04
25000	11,17	3,30	4,21	3,93	1,89	2,37	2,36	6,61
50000	11,97	3,61	4,60	4,04	2,02	2,53	2,52	7,19
100000	12,88	3,94	5,01	4,18	2,18	2,72	2,72	7,89
200000	13,89	4,26	5,45	4,30	2,32	2,90	2,90	8,56
400000	15,40	4,61	5,85	4,43	2,45	3,06	3,06	9,26
800000	17,30	4,93	6,28	4,57	2,58	3,24	3,24	9,96
1600000	19,41	5,26	6,71	4,68	2,73	3,43	3,43	10,67
3200000	21,76	5,64	7,20	4,87	2,90	3,63	3,63	11,47
6400000	24,04	5,97	7,61	5,00	3,02	3,80	3,81	12,17
12800000	26,38	6,31	8,03	5,14	3,15	3,98	3,99	12,83

Ek 8 Sıralama Uygulamaları Test Sonuçları

8.1 msort_TD Sıralama Algoritmasının C Diliyle Gerçekleştirimi

```
void msort_td(int a[],int b[],int dd,int hh){
        int i;
        for(i=dd;i<=hh;i++) b[i] = a[i];
        msortAB_td(a,b,dd,hh);
}
void msortAB_td(int a[],int b[],int dd,int hh) {
        int dh = (dd+hh)/2;
        if(hh-dd <= KESME2) {InsertionSort(a,dd,hh); return;}
        msortAB_td(b,a,dd,dh);
        msortAB_td(b,a,dh+1,hh);
        mergeAB(&a[dd],&b[dd],dh-dd+1,&b[dh+1],hh-dh);
}
void mergeAB(int c[], int a[],int m,Item b[] ,int n) {
        int i,j,k;
        for(i=0,j=0,k=0;k<n+m;k++) {
            if(i==m) {c[k] = b[j++]; continue; }
            if(j==n) {c[k] = a[i++]; continue; }
            c[k] = (less(a[i],b[j])) ? a[i++] : b[j++];
}}
void InsertionSort(int a[],int dd,int hh) {
        int i,j,ek_alan;
        for (i =hh ; i> dd ; i--)
            if(a[i] < a[i-1]) {ek_alan=a[i];a[i]=a[i-1];a[i-1]=ek_alan;}
        for (i=dd+2; i<=hh;i++) {
            j=i; ek_alan=a[i];
            while( ek_alan<a[j-1] )
                {a[j] = a[j-1] ; j--; }
        a[j] = ek_alan;
}}
```

8.2 Sıralama Uygulamalarının Test Süreleri (mS)

	N	msort	msort_ATY1	msort_ATY2	msort_ATD1	msort_ATD2
KESME1=16 & KESME2=16	12500	5	15,86	13,27	12,66	10,3
	25000	10,7	36,79	30,31	29,29	23,37
	50000	22,9	84,4	68,75	66,88	52,5
	100000	48,7	191,69	154,61	152,82	117,33
	200000	103,18	432,6	345,59	344,34	263,17
	400000	218,00	969,57	766,92	770,90	579,65
	800000	459,23	2156,49	1693,91	1711,11	1270,76
	1600000	964,86	4772,80	3721,35	3777,69	2774,43
	3200000	2023,45	10519,80	8148,98	8307,24	6036,19
	6400000	4235,07	23071,13	17773,25	18182,94	13100,03
	12800000	8843,89	50428,25	38603,82	39647,04	28249,92
KESME1=16 & KESME2=32	12500	5,08	15,85	13,24	12,79	10,43
	25000	10,88	36,81	30,26	29,53	23,59
	50000	23,23	84,32	68,65	67,43	52,99
	100000	49,45	191,56	154,69	153,03	119,32
	200000	104,66	432,12	345,19	347,05	264,09
	400000	221,12	968,43	767,16	774,36	583,94
	800000	464,84	2154,62	1692,79	1720,29	1279,22
	1600000	976,30	4772,56	3723,78	3792,13	2789,82
	3200000	2046,05	10513,89	8146,38	8337,36	6068,36
	6400000	4282,55	23100,06	17774,65	18249,14	13128,44
	12800000	8938,11	50458,36	38578,15	39755,46	28364,15
KESME1=32 & KESME2=32	12500	5	13,51	11,39	10,91	8,94
	25000	10,71	31,51	26,12	25,33	20,29
	50000	22,88	72,64	59,47	58,13	45,61
	100000	48,73	166,17	134,5	132,69	102,41
	200000	103,16	377,23	302,31	304,35	230,37
	400000	218,07	849,14	674,2	681,49	510,31
	800000	459,43	1899,61	1496,33	1517,19	1122,66
	1600000	966,62	4227,52	3298,74	3361,64	2458,94
	3200000	2027,53	9355,79	7249,91	7419,83	5364,57
	6400000	4236,62	20601,7	15853,04	16313,58	11650,43
	12800000	8844,79	45215,21	34613,07	35625,5	25234,07

8.3 Sıralama Uygulamalarının Test Sürelerinin Normalleştirilmiş Değerleri

	N	msort_ATY1	msort_ATY2	msort_ATD1	msort_ATD2
KESME1=16 & KESME2=16	12500	3,17	2,65	2,53	2,06
	25000	3,44	2,83	2,74	2,18
	50000	3,69	3,00	2,92	2,29
	100000	3,94	3,17	3,14	2,41
	200000	4,19	3,35	3,34	2,55
	400000	4,45	3,52	3,54	2,66
	800000	4,70	3,69	3,73	2,77
	1600000	4,95	3,86	3,92	2,88
	3200000	5,20	4,03	4,11	2,98
	6400000	5,45	4,20	4,29	3,09
	12800000	5,70	4,37	4,48	3,19
KESME1=16 & KESME2=32	12500	3,12	2,61	2,52	2,05
	25000	3,38	2,78	2,71	2,17
	50000	3,63	2,96	2,90	2,28
	100000	3,87	3,13	3,09	2,41
	200000	4,13	3,30	3,32	2,52
	400000	4,38	3,47	3,50	2,64
	800000	4,64	3,64	3,70	2,75
	1600000	4,89	3,81	3,88	2,86
	3200000	5,14	3,98	4,07	2,97
	6400000	5,39	4,15	4,26	3,07
	12800000	5,65	4,32	4,45	3,17
KESME1=32 & KESME2=32	12500	2,70	2,28	2,18	1,79
	25000	2,94	2,44	2,37	1,89
	50000	3,17	2,60	2,54	1,99
	100000	3,41	2,76	2,72	2,10
	200000	3,66	2,93	2,95	2,23
	400000	3,89	3,09	3,13	2,34
	800000	4,13	3,26	3,30	2,44
	1600000	4,37	3,41	3,48	2,54
	3200000	4,61	3,58	3,66	2,65
	6400000	4,86	3,74	3,85	2,75
	12800000	5,11	3,91	4,03	2,85

Ek 9 Paralel Algoritmaların Benzetim Test Sonuçları

9,1 Test Süreleri (μS)

P	N	m/N=0.1		m/N=0.25		m/N=0.5		m/N=0.75		m/N=0.9	
		ATP1	ATP2	ATP1	ATP2	ATP1	ATP2	ATP1	ATP2	ATP1	ATP2
2	12500	576,93	884,27	791,93	1021,32	1018,02	934,86	807,18	1041,45	588,05	889,86
	25000	1188,16	1835,55	1660,69	2194,53	2108,78	1982,94	1666,23	2215,98	1211,50	1870,36
	50000	2535,96	3916,87	3443,59	4673,68	4418,12	4205,08	3462,14	4720,19	2556,57	3987,28
	100000	5356,91	8322,16	7206,12	9820,73	9319,95	8815,75	7242,50	9912,35	5460,27	8615,40
	200000	11544,85	18165,97	15061,97	20750,05	19684,94	18564,12	15180,77	20944,98	11659,37	18342,90
	400000	23978,38	37887,30	31766,31	44154,88	40987,36	38802,14	31880,56	44341,62	24357,39	38401,12
	800000	49950,35	78846,80	66527,01	91793,85	85877,32	81608,38	66849,91	92779,42	50600,32	79678,45
	1600000	104996,28	167285,58	139113,31	192949,16	179204,78	170133,07	142833,62	194329,37	105605,36	169005,16
	3200000	220006,64	350240,85	289293,25	403156,59	373239,63	355317,04	291090,22	407501,68	220995,73	354863,78
	6400000	458026,67	734320,93	602094,21	845217,36	776824,22	736479,56	604097,71	847905,55	461860,68	741194,11
	12800000	948635,43	1533333,68	1251572,84	1761259,07	1622851,09	1540208,24	1258762,35	1770412,39	959719,60	1548195,43
4	12500	429,40	852,96	540,06	854,94	628,68	546,91	545,27	807,45	437,95	868,20
	25000	873,89	1740,83	1078,87	1737,72	1294,22	1114,82	1089,12	1712,91	879,77	1728,86
	50000	1826,41	3626,57	2201,33	3631,13	2710,92	2352,41	2219,46	3650,02	1842,14	3646,27
	100000	3718,91	7706,45	4618,44	7712,23	5648,66	4927,70	4661,26	7714,25	3730,99	7808,10
	200000	7901,46	16491,24	9591,03	16250,09	11798,80	10310,24	9686,88	16399,65	7913,06	16778,01
	400000	16090,34	34450,15	20155,63	34311,21	24509,63	21319,25	20355,98	34288,79	16275,63	34907,38
	800000	33424,11	71713,29	41873,22	71696,38	51351,00	44472,38	42424,97	72114,06	33595,81	73332,05
	1600000	69458,96	150706,93	86914,30	150026,29	107036,04	92099,79	88972,83	152838,52	70152,23	154508,94
	3200000	145031,59	317989,57	180622,58	312293,75	222411,97	190156,85	181429,11	315216,07	146492,99	324300,15
	6400000	302080,18	661880,98	374865,13	650195,20	461912,56	394369,79	375260,94	654551,69	303150,69	676111,09
	12800000	631925,75	1383212,93	771698,77	1349133,59	955436,95	819948,92	778464,30	1364795,87	635489,51	1414669,88
8	12500	510,87	822,65	556,35	853,47	579,31	497,27	616,94	767,29	516,82	822,09
	25000	922,51	1694,59	1015,62	1697,70	1078,67	952,79	1147,20	1535,18	911,61	1685,63
	50000	1768,51	3591,33	2006,53	3385,82	2144,81	1815,52	2113,07	3119,19	1792,74	3516,26
	100000	3447,27	7519,96	3984,90	6825,93	4242,84	3438,57	3979,05	6399,21	3589,60	7390,38
	200000	6905,08	15876,54	7607,53	13811,03	8506,22	6714,64	7648,49	13197,75	7234,20	15682,88
	400000	13109,67	32188,86	15025,72	27839,64	16989,30	13217,46	14929,08	27472,21	13696,02	32387,94
	800000	26101,72	67537,88	30320,83	58064,32	34074,09	27187,12	29874,15	56964,15	26623,88	67449,59
	1600000	52677,12	142663,65	61169,61	121783,99	70754,01	55650,14	62971,04	120003,18	53894,80	141785,35
	3200000	107319,38	301765,00	126641,21	252818,82	143710,73	113015,28	123300,67	246786,50	110482,26	295951,32
	6400000	219310,18	631518,68	256666,44	527114,49	296441,81	236069,87	259181,76	511162,58	221139,72	617763,38
	12800000	449705,54	1320919,19	523842,55	1097009,70	617176,64	491513,71	527475,05	1063362,90	455054,85	1291222,29

9,2 Hızlanma

P	N	m/N=0.1		m/N=0.25		m/N=0.5		m/N=0.75		m/N=0.9	
		ATP1	ATP2	ATP1	ATP2	ATP1	ATP2	ATP1	ATP2	ATP1	ATP2
2	12500	1,58	1,03	1,66	1,29	1,76	1,92	1,66	1,28	1,58	1,05
	25000	1,61	1,05	1,70	1,29	1,83	1,95	1,71	1,29	1,63	1,06
	50000	1,63	1,06	1,74	1,28	1,86	1,96	1,76	1,29	1,66	1,06
	100000	1,67	1,07	1,78	1,30	1,86	1,97	1,78	1,30	1,67	1,06
	200000	1,68	1,07	1,79	1,30	1,86	1,97	1,80	1,30	1,69	1,08
	400000	1,69	1,07	1,81	1,30	1,87	1,97	1,80	1,29	1,69	1,07
	800000	1,71	1,08	1,81	1,31	1,87	1,97	1,81	1,30	1,71	1,09
	1600000	1,71	1,08	1,82	1,31	1,87	1,97	1,82	1,34	1,74	1,09
	3200000	1,73	1,08	1,83	1,31	1,88	1,97	1,83	1,30	1,75	1,09
	6400000	1,74	1,08	1,84	1,31	1,88	1,98	1,84	1,31	1,75	1,09
	12800000	1,76	1,09	1,84	1,31	1,88	1,98	1,84	1,31	1,76	1,09
4	12500	2,12	1,07	2,44	1,54	2,86	3,28	2,45	1,66	2,13	1,07
	25000	2,20	1,10	2,62	1,63	2,99	3,47	2,62	1,67	2,24	1,14
	50000	2,27	1,14	2,73	1,65	3,04	3,50	2,75	1,67	2,30	1,16
	100000	2,40	1,16	2,77	1,66	3,07	3,52	2,77	1,67	2,45	1,17
	200000	2,46	1,18	2,81	1,66	3,10	3,55	2,82	1,67	2,49	1,18
	400000	2,52	1,18	2,85	1,67	3,12	3,59	2,82	1,67	2,53	1,18
	800000	2,55	1,19	2,88	1,68	3,13	3,61	2,85	1,68	2,58	1,18
	1600000	2,59	1,19	2,91	1,69	3,14	3,64	2,93	1,70	2,62	1,19
	3200000	2,62	1,19	2,93	1,69	3,15	3,69	2,93	1,69	2,64	1,19
	6400000	2,63	1,20	2,95	1,70	3,16	3,71	2,96	1,70	2,67	1,20
	12800000	2,64	1,20	2,98	1,71	3,19	3,72	2,98	1,70	2,66	1,20
8	12500	1,78	1,11	2,37	1,54	3,10	3,61	2,17	1,74	1,80	1,13
	25000	2,08	1,13	2,78	1,66	3,59	4,06	2,49	1,86	2,16	1,17
	50000	2,34	1,15	2,99	1,77	3,84	4,53	2,89	1,96	2,36	1,21
	100000	2,59	1,19	3,21	1,87	4,09	5,04	3,25	2,02	2,54	1,24
	200000	2,81	1,22	3,54	1,95	4,30	5,45	3,57	2,07	2,73	1,26
	400000	3,09	1,26	3,82	2,06	4,50	5,79	3,84	2,09	3,00	1,27
	800000	3,27	1,26	3,98	2,08	4,72	5,91	4,05	2,12	3,26	1,29
	1600000	3,42	1,26	4,13	2,08	4,74	6,03	4,13	2,17	3,40	1,29
	3200000	3,54	1,26	4,17	2,09	4,88	6,20	4,31	2,15	3,49	1,30
	6400000	3,62	1,26	4,31	2,10	4,93	6,19	4,29	2,18	3,66	1,31
	12800000	3,70	1,26	4,39	2,10	4,94	6,21	4,40	2,18	3,72	1,31

9,3 Verim

P	N	m/N=0.1		m/N=0.25		m/N=0.5		m/N=0.75		m/N=0.9	
		ATP1	ATP2	ATP1	ATP2	ATP1	ATP2	ATP1	ATP2	ATP1	ATP2
2	12500	0,79	0,52	0,83	0,64	0,88	0,96	0,83	0,64	0,79	0,52
	25000	0,81	0,52	0,85	0,64	0,92	0,98	0,86	0,64	0,81	0,53
	50000	0,82	0,53	0,87	0,64	0,93	0,98	0,88	0,65	0,83	0,53
	100000	0,83	0,54	0,89	0,65	0,93	0,98	0,89	0,65	0,84	0,53
	200000	0,84	0,54	0,89	0,65	0,93	0,99	0,90	0,65	0,85	0,54
	400000	0,85	0,54	0,90	0,65	0,93	0,99	0,90	0,65	0,84	0,54
	800000	0,85	0,54	0,91	0,66	0,94	0,98	0,90	0,65	0,86	0,54
	1600000	0,86	0,54	0,91	0,66	0,94	0,99	0,91	0,67	0,87	0,54
	3200000	0,86	0,54	0,91	0,66	0,94	0,99	0,91	0,65	0,87	0,54
	6400000	0,87	0,54	0,92	0,65	0,94	0,99	0,92	0,66	0,88	0,55
	12800000	0,88	0,54	0,92	0,65	0,94	0,99	0,92	0,66	0,88	0,55
4	12500	0,53	0,27	0,61	0,39	0,71	0,82	0,61	0,41	0,53	0,27
	25000	0,55	0,28	0,65	0,41	0,75	0,87	0,66	0,42	0,56	0,29
	50000	0,57	0,29	0,68	0,41	0,76	0,87	0,69	0,42	0,58	0,29
	100000	0,60	0,29	0,69	0,41	0,77	0,88	0,69	0,42	0,61	0,29
	200000	0,62	0,29	0,70	0,41	0,78	0,89	0,70	0,42	0,62	0,29
	400000	0,63	0,29	0,71	0,42	0,78	0,90	0,70	0,42	0,63	0,29
	800000	0,64	0,30	0,72	0,42	0,78	0,90	0,71	0,42	0,65	0,30
	1600000	0,65	0,30	0,73	0,42	0,78	0,91	0,73	0,43	0,65	0,30
	3200000	0,65	0,30	0,73	0,42	0,79	0,92	0,73	0,42	0,66	0,30
	6400000	0,66	0,30	0,74	0,43	0,79	0,93	0,74	0,42	0,67	0,30
	12800000	0,66	0,30	0,75	0,43	0,80	0,93	0,75	0,43	0,67	0,30
8	12500	0,22	0,14	0,30	0,19	0,39	0,45	0,27	0,22	0,23	0,14
	25000	0,26	0,14	0,35	0,21	0,45	0,51	0,31	0,23	0,27	0,15
	50000	0,29	0,14	0,37	0,22	0,48	0,57	0,36	0,24	0,30	0,15
	100000	0,32	0,15	0,40	0,23	0,51	0,63	0,41	0,25	0,32	0,15
	200000	0,35	0,15	0,44	0,24	0,54	0,68	0,45	0,26	0,34	0,16
	400000	0,39	0,16	0,48	0,26	0,56	0,72	0,48	0,26	0,38	0,16
	800000	0,41	0,16	0,50	0,26	0,59	0,74	0,51	0,27	0,41	0,16
	1600000	0,43	0,16	0,52	0,26	0,59	0,75	0,52	0,27	0,43	0,16
	3200000	0,44	0,16	0,52	0,26	0,61	0,78	0,54	0,27	0,44	0,16
	6400000	0,45	0,16	0,54	0,26	0,62	0,77	0,54	0,27	0,46	0,16
	12800000	0,46	0,16	0,55	0,26	0,62	0,78	0,55	0,27	0,46	0,16

Ek 10 Türkçe-İngilizce Terimler Sözlüğü

Türkçe	**İngilizce**
ayrıştırma tabanlı	decomposition-based
bellek	memory
benzetim	simulation
birleştirme	merge
bölümleme	partition
çalıştırma ortamı	run-time environment
dağıtık	distributed
değiş tokuş	swap
dışsal birleştirme	external merge
doğrusal	linear
en aza indirgemek	minimize
en iyi	optimum
en iyileme	optimization
gerçekleştirim	implementation
gerçekleştirmek	to implement
hızlanma	speed-up
içsel birleştirme	internal merge
imleç	pointer
işletmen	operator
kaba kod	pseudocode
kaçırma oranı	miss rate
kararlı	stable
kararsız	instable
karmaşıklık	complexity

kesim	segment
kesmek	cut off
küme	(1)set (2)cluster
özyinelemeli	recursive
taşıma	movement
ters permütasyon	inverse permutation
uyarlamalı	adaptive
uygulama	application
varolan	existing
yardımcı	auxiliary
yarar	advantage
yerinde	in-place

ÖZGEÇMİŞ

İlker KOCABAŞ

adres: 124 SK, No:3/1 K:10 D:20 Evka-3 / İZMİR

tel: (0532) 3246905

e-mail: ikocabas@ube,ege,edu,tr

Kişisel Bilgiler	Milliyeti	: T,C,
	Doğum Yeri / Tarihi	: İzmir / 25,04,1978

Eğitim	2001 -	Ege Üniversitesi Uluslararası Bilgisayar Enstitüsü (UBE) *Bilgi Teknolojileri Ana Bilim Dalı*
	1995 – 2000	Orta Doğu Teknik Üniversitesi Mühendislik Fakültesi Elektrik ve Elektronik Mühendisliği Bölümü

Mesleki İlgi Alanları	Dağıtık/Paralel sistemler
	Gömülü sistemler
	Yazılım sınama ve doğrulama
	Yazılım sistem mimarisi ve tasarım desenleri
	Veri tabanı tasarımı ve yönetimi