

EGE ÜNİVERSİTESİ FEN BİLİMLERİ ENSTİTÜSÜ
(DOKTORA TEZİ)

ÖRGÜ SİSTEMLERDE SORGU İŞLEMESİ İÇİN
KAYNAK TARAMASI VE ATAMASI

Deniz ÇOKUSLU

Tez Danışmanı: Prof. Dr. Kayhan ERCİYEŞ

Uluslararası Bilgisayar Anabilim Dalı

Bilim Dalı Kodu: 619.02.04.
Sunuş Tarihi: 02.11.2012

Bornova-İZMİR

2012

Deniz OKUSLU tarafından doktora tezi olarak sunulan “Örgü Sistemlerde Sorgu İşlemesi İçin Kaynak Taraması ve Ataması” başlıklı bu çalışma E.Ü. Lisansüstü Eğitim ve Öğretim Yönetmeliğı ile E.Ü. Fen Bilimleri Enstitüsü Eğitim ve Öğretim Yönergesi’nin ilgili hükümleri uyarınca tarafımızdan değerlendirilerek savunmaya değer bulunmuş ve 02.11.2012 tarihinde yapılan tez savunma sınavında aday oybirliğı ile başarılı bulunmuştur.

Jüri Üyeleri:

İmza

Jüri Başkanı : Prof. Dr. Kayhan ERCİYEŞ

Raportör Üye : Doç. Dr. Muhammed CİNSDİKİCİ

Üye : Prof. Dr. Mehmet Emin DALKILIÇ

Üye : Prof. Dr. Nihan Kesim ÇİÇEKLİ

Üye : Doç. Dr. Aylin KANTARCI

ÖZET**ÖRGÜ SİSTEMLERDE SORGU İŞLEMESİ İÇİN KAYNAK
TARAMASI VE ATAMASI**

ÇOKUSLU, Deniz

Doktora Tezi, Uluslararası Bilgisayar Anabilim Dalı

Tez Yöneticisi: Prof. Dr. Kayhan ERCİYEŞ

Kasım 2012, 151 sayfa

Örgü sistemleri geniş ölçekli hesaplama kabiliyetleri ve yüksek erişilebilirlikleri sebebiyle günümüzün ilgi çeken araştırma konularından biri haline gelmiştir. Bu sebeple birçok güncel araştırma alanı örgü sistemlerinin sunduğu olanaklardan faydalanmaktadır. Dağıtık sorgu işleme, mevcut altyapısını koştur ve dağıtık sistemlerden örgü sistemlere taşımak amacıyla yüksek miktarda araştırmalar yapılan alanlardan birisidir.

Bu tezde, örgü sistemler üzerinde sorgu işlemesi için kaynak taraması ve kaynak ataması konuları üzerinde odaklanılmaktadır. Bunun için, öncelikle örgü sistemleri üzerinde sorgu işlemesi için öz-kararlı ilinge denetimi ve kapsayan ağaç tabanlı kaynak taraması algoritmaları önerilmektedir. Daha sonra tek birleştirme işletmenli sorguların örgü sistemleri üzerinde işletilmesi için, küçültülmüş aday düğüm arama uzayı ve aday düğümlerin veri kaynaklarına olan yakınlıklarını kullanan kaynak atama algoritmaları önerilmektedir. Önerilen bu kaynak atama algoritması, çoklu birleştirme işletmenli sorgularda kullanılabilirlik üzere ilerletilmiş ve daha sonra da hataya dayanıklı olacak şekilde geliştirilmiştir.

Bu tezin, bilime çift taraflı olarak katkı koyması planlanmıştır. İlk olarak, örgü sistemlerinin özellikleri göz önüne alınarak kaynak tarama algoritmaları önerilmektedir. Örgü sistemlerdeki ölçeklenebilirlik ve devingenlik problemleri, örgü üzerinde oluşturulan öz kararlı ilinge denetimi algoritmaları ile çözülmektedir. Sistemdeki kaynakların türdeş olmamasından kaynaklanan problemler ise kapsayan ağaç tabanlı kaynak tarama algoritmaları ile çözümlenmektedir. Örgü sistemlerinin özellikleri gözetilerek önerilen kaynak atama algoritmaları ise tezin ikinci katkısını oluşturmaktadır. Bu aşamada, ölçeklenebilirlik problemleri aday düğüm arama uzayının boyutunun küçültülmesi ile çözülmektedir. Sorgunun işletilmesi sırasında iletişim maliyetlerinin

azaltılması amacıyla veri kaynaklarına yakın olan düğümün sorgu işlemlerine katılması sağlanmaktadır. Son olarak örgü sistemindeki düğümün devingenliklerinden kaynaklanan problemlerin çözümü için hataya dayanıklı kaynak atama algoritması önerilmektedir.

Anahtar sözcükler: Sorgu işleme, kaynak taraması, öz-kararlı algoritmalar, ilinge denetimi, kapsayan ağaç, kaynak atama, hataya dayanıklı algoritmalar.

ABSTRACT**RESOURCE DISCOVERY AND ALLOCATION FOR QUERY
PROCESSING IN GRID SYSTEMS**

ÇOKUSLU, Deniz

PhD in International Computer Department

Supervisor: Prof. Dr. Kayhan ERCİYEŞ

November 2012, 151 pages

Grid systems are today's one of the most interesting computing environments because of their large computing and storage capabilities and their availability. Many different domains profit the facilities of grid environments. Distributed query processing is one of these domains in which there exists large amounts of ongoing research to port the underlying environment from distributed and parallel systems to the grid environment.

In this thesis, we focus on resource discovery and resource allocation algorithms for query processing in grid environments. For this, we propose resource discovery algorithm for query processing in grid systems by introducing self-stabilizing topology control and converge-cast based resource discovery algorithms. Then, we propose a resource allocation algorithm, which realizes allocation of resources for single join operator queries by generating a reduced search space for the candidate nodes and by considering proximities of candidates to the data sources. We also propose another resource allocation algorithm for queries with multiple join operators. Lastly, we propose a fault-tolerant resource allocation algorithm, which provides fault-tolerance during the execution of the query by the use of passive replication of stateful operators.

The general contribution of this thesis is twofold. First, we propose a new resource discovery algorithm by considering the characteristics of the grid environments. We address scalability and dynamicity problems by constructing an efficient topology over the grid environment using the self-stabilization concept; and we deal with the heterogeneity problem by proposing the converge-cast based resource discovery algorithm. The second main contribution of this thesis is the proposition of a new resource allocation algorithm considering the characteristics of the grid environment. We tackle the scalability problem by reducing the search

space for candidate resources. We decrease the communication costs during the query execution by allocating nodes closer to the data sources. And finally we deal with the dynamicity of nodes, in terms of their existence in the system, by proposing the fault-tolerant resource allocation algorithm.

Keywords: Query processing, resource discovery, self-stabilization, topology control, spanning tree, resource allocation, fault-tolerance.

TEŞEKKÜR

Bu tezin hazırlanması süresince bana yol gösteren değerli tez danışmanlarım Prof. Dr. Kayhan Erciyeş ve Prof. Dr. Abdelkader Hameurlain'e derin teşekkürlerimi sunarım.

İYTE Bilgisayar Mühendisliği Bölümü bölüm başkanı Prof. Dr. Sıtkı Aytaç'a, Ege Üniversitesi Uluslararası Bilgisayar Enstitüsü başkanı Prof. Dr. Mehmet Emin Dalkılıç'a ve Paul Sabatier Üniversitesi IRIT Laboratuvarı başkanı Prof. Dr. Michel Daydé'ye doktora çalışmalarım boyunca bana verdikleri destekten dolayı teşekkürlerimi sunarım.

Doç. Dr. Aylin Kantarcı'ya ve Prof. Dr. Mehmet Emin Dalkılıç'a da tez izleme komitemde bulunmayı kabul ettikleri ve tez izleme komitesi toplantılarında sundukları değerli görüşleri için derin teşekkürlerimi sunarım.

Türkiye Bilimsel ve Teknolojik Araştırma Kurumu TÜBİTAK'a yurtiçi doktora burs programından faydalanmama olanak sağladığı için teşekkürlerimi sunarım. Benzer şekilde, tezimle ilgili olarak Fransa'da yürüttüğüm çalışmalara destek sağladığı için de Fransız Hükümeti'ne teşekkürlerimi borç bilirim.

Doktora çalışmalarım süresince yanımda olan ve bana destek olan tüm dostlarıma teşekkürlerimi sunarım. Özellikle tezimi hazırladığım süre boyunca bana içtenlikle inanan ve bana destek olan Selçuk Sözen'e teşekkürlerimi borç bilirim.

Aileme bana destek oldukları, iyisiyle kötüsüyle her anımda yanımda oldukları için en içten ve en derin teşekkürlerimi sunarım. Son olarak, eşim Evşen'e beni cesaretlendirdiği ve destek olduğu için derin teşekkürlerimi sunarım.

İÇİNDEKİLER

	<u>Sayfa</u>
ÖZET.....	V
ABSTRACT.....	Vİİ
TEŞEKKÜR.....	İX
İÇİNDEKİLER.....	Xİ
ŞEKİLLER DİZİNİ.....	XVİİ
ÇİZELGELER DİZİNİ.....	XİX
1. GİRİŞ.....	1
1.1 Kapsam, Motivasyon ve Problem Tanımı.....	1
1.2 Örgü Sistemlerde Sorgu İşleme için Kaynak Taraması.....	3
1.3 Örgü Sistemlerde Sorgu İşleme için Kaynak Ataması.....	4
1.4 Deneysel Doğrulama.....	6
1.5 Katkılar.....	7
1.6 Tezin Yapısı.....	9
2. LİTERATÜR ÖZETİ.....	10
2.1 Giriş.....	10

İÇİNDEKİLER (devam)

	<u>Sayfa</u>
2.2 Örgü Sistemlerde Sorgu İşlemesi için Kaynak Tarama Algoritmaları.....	11
2.2.1 Merkezi ve sıradüzensel mimariler tabanlı kaynak tarama.....	13
2.2.2 Eşler arası sistemler (P2P) tabanlı örgü kaynak taraması.....	17
2.2.3 Etmen sistemler tabanlı örgü kaynak taraması.....	24
2.2.4 Kaynak tarama yöntemlerinin genel değerlendirilmesi.....	28
2.2.5 Öz-kararlı kapsayan ağaç oluşturma algoritmaları.....	30
2.2.6 Kaynak taraması algoritmaları için sonuçlar.....	33
2.3 Örgü Sistemleri Üzerinde Sorgu İşlemesi İçin Kaynak Atama Algoritmaları.....	33
2.3.1 Mevcut literatür özeti çalışmaları ve motivasyonlar.....	36
2.3.2 Yakın zamanda yapılan kaynak atama çalışmaları.....	37
2.3.3 Örgü sistemlerde sorgu işlemede hataya dayanıklılık.....	47
2.3.4 Kaynak atama algoritmaları için sonuçlar.....	49
2.4 Genel Sonuçlar.....	50
3. ÖRGÜ SİSTEMLERDE SORGU İŞLEME İÇİN KAYNAK TARAMASI.....	53

İÇİNDEKİLER (devam)

	<u>Sayfa</u>
3.1 Giriş.....	53
3.2 Örgü Sistemlerde Kaynak Taraması için İlinge Denetimi.....	54
3.2.1 Karşılaştırma amaçlı seçilen algoritmalar.....	56
3.2.2 Önerilen algoritmalar.....	58
3.2.3 Analizler.....	64
3.2.4 Benzetimler.....	66
3.2.5 Sonuç.....	70
3.3 Örgü Sistemlerde Sorgu İşleme İçin Kapsayan Ağaç Tabanlı Kaynak Tarama.....	70
3.3.1 Kapsayan ağaç tabanlı kaynak tarama algoritması.....	71
3.3.2 Analizler	72
3.3.3 Benzetimler.....	73
3.3.4 Sonuçlar.....	74
3.4 Genel Sonuçlar.....	75
4. ÖRGÜ SİSTEMLERDE SORGU İŞLEME İÇİN KAYNAK ATAMASI.....	76

İÇİNDEKİLER (devam)

	<u>Sayfa</u>
4.1 Giriş.....	77
4.2 Tekli Birleştirme İşletmeni Kaynak Atama (TBİKA) Algoritması	79
4.2.1 Yakınlık tabanlı aday düğüm listesi oluşturma (YADLO) algoritması.....	80
4.2.2 Birleştirme görevi kaynak atama (BGKA) algoritması.....	82
4.2.3 Analizler.....	86
4.2.4 Benzetimler.....	87
4.2.5 Sonuç.....	89
4.3 Çoklu Birleştirme İşletmenli Sorgular için Kaynak Atama Algoritması.....	90
4.3.1 Çoklu birleştirme işletmeni kaynak atama (ÇBİKA) algoritması.....	90
4.3.2 Analizler.....	93
4.3.3 Benzetimler.....	93
4.3.4 Sonuç.....	96
4.4 Genel Sonuçlar.....	96
5. ÖRGÜ SİSTEMLERDE SORGU İŞLEME İÇİN HATAYA DAYANIKLI KAYNAK ATAMA.....	97

İÇİNDEKİLER (devam)

	<u>Sayfa</u>
5.1 Giriş.....	98
5.2 Hataya Dayanıklı Kaynak Atama (HDKA) Algoritması.....	100
5.2.1 Algoritma.....	100
5.2.2 Analizler.....	108
5.2.3 Benzetimler.....	109
5.3 Genel Sonuçlar.....	110
6. SONUÇLAR.....	111
6.1 Çalışmaların Özeti.....	112
6.2 Gelecekte Yapılması Planlanan İşler.....	114
6.3 Genel Sonuçlar.....	115
KAYNAKLAR DİZİNİ.....	117
ÖZGEÇMİŞ.....	129

ŞEKİLLER DİZİNİ

<u>Şekil</u>	<u>Sayfa</u>
2.1 Örgü sistemlerinde sorgu işleme için kaynak taraması algoritmaları sınıflandırması.....	13
2.2 Örgü sistemlerinde sorgu işleme için kaynak atama yöntemleri sınıflandırma özeti.....	34
2.3 Değişen ortam büyüklüklerine göre kaynak atama maliyetleri.....	45
2.4 Değişen düğüm sayılarına göre farklı görev yükleri için başarımlar kazanımları.....	45
2.5 Değişen işlemci yüklerine göre görev tamamlanma süreleri.....	46
2.6 Değişen düğüm sayılarına göre görevlerin tamamlanma süreleri...	47
3.1 EDKA1 Algoritmasının sonlu konum diyagramı.....	61
3.2 Kapsayan ağaç oluşturma algoritmaları çalışma süreleri.....	68
3.3 Kapsayan ağaç oluşturma algoritmalarında elde edilen kapsayan ağaçların çapları.....	69
3.4 KAKT Algoritmasının akış diyagramı.....	72
3.5 Kaynak tarama algoritmaları tamamlanma süreleri.....	74
4.1 Kaynak ataması için gerekli süreler.....	88
4.2 Sorgu işleme süreleri.....	89
4.3 Örnek bir sorgunun ağaç gösterimi.....	91

ŞEKİLLER DİZİNİ (devam)

<u>Şekil</u>	<u>Sayfa</u>
4.4. Sorgu ağacının köşelerini temsil eden veri yapısı.....	92
4.5 Örnek sorgunun biçimsel gösterimi.....	94
4.6 Kaynak ataması için gereken süreler.....	95
4.7 Örnek sorgu çalışma süreleri.....	96
5.1 HDKA Algoritmasına girdi olarak sunulan sorgu ağaç gösterimi..	102
5.2 Sorgu ağacının köşelerini temsil eden veri yapısı.....	102
5.3 HDKA Algoritmasının sonlu durum makinesi.....	106
5.4 HDKA algoritması kaynak atama süreleri.....	110

ÇİZELGELER DİZİNİ

<u>Çizelge</u>	<u>Sayfa</u>
2.1 Merkezi ve sıradüzensel kaynak tarama yöntemleri sınıfları arasında karşılaştırma.....	16
2.2 Eşler-arası sistemler tabanlı kaynak tarama yöntemleri karşılaştırma özeti.....	22
2.3 Etmen sistemler tabanlı kaynak tarama yöntemleri sınıflar arası karşılaştırma özeti.....	27
2.4 Farklı kaynak tarama yöntemi sınıfları arasında karşılaştırma özeti.....	29
2.5 Kaynak Atama Yöntemleri Sınıflar Arası Niteliksel Karşılaştırma Özeti.....	42

1.GİRİŞ

1.1 Kapsam, Motivasyon ve Problem Tanımı

Araştırma aktiviteleri yeni uygulamalar, teknoloji eğilimleri ve yenilikçi çalışmalar tarafından yönlendirilmektedir. 90'lı yıllardan bu yana internet, bilimsel yazılım geliştirme konusunda önemli güdüleyici etmenlerden biri haline gelmiştir. Teknolojik gelişim sürdükçe, hızla büyüyen miktarlardaki verinin yönetilebilmesi için Veritabanı Yönetim Sistemleri (VTYS) gibi karmaşık sistemler geliştirilmiştir. Bu sistemler büyüyen veri erişim ihtiyacını hızlı ve güvenilir şekilde karşılayacak yöntemler sunmaktadırlar. İlk VTYS'ler merkezi sunucular üzerinde çalışacak şekilde tasarlanmışlardır (DeWitt, et al., 1984). Ancak, kullanıcıların ve verinin ölçeğinin büyümesiyle merkezi VTYS'ler yetersiz kalmışlardır. Bu problemi çözebilmek için araştırmacılar dikkatlerini Dağıtık Veritabanı Yönetim Sistemleri'ne (DVTYS) çevirmişlerdir (Özsu and Valduries, 2011). VTYS'yi ağ üzerinde dağıtık hale getirmek bilgi teknolojilerine esneklik ve ölçeklenebilirlik imkânı sunmuştur. Verinin dağıtılmasından sonra araştırmacılar, bu sistemlerin esnekliklerini arttırmak amacıyla ilgilerini DVTYS, XML, Dosya Sistemleri gibi değişik veri kaynağı tiplerini tümleştirmeye çevirmişlerdir. Bu oluşum, geniş ölçekli sistemlerde türdeş olmayan veri kaynaklarını tümleştiren sistemler olarak tanımlanabilen arabulucu sistemi kavramının oluşmasına sebep olmuştur (Özsu and Valduries, 2011).

Bilgi teknolojilerindeki ilerleme, beraberinde geniş ölçekli sistemlerdeki paylaşılan veriye ulaşım ihtiyacını da büyötmüştür. Bu durum araştırmacıların Örgü sistemleri adındaki birbirine bağı sanal örgütleri geliştirmelerine önayak olmuştur. Örgü sistemleri kullanıcılarına geniş ölçekli ve güçlü hesaplama olanakları sağlayan dağıtık hesaplama ortamları olarak tanımlanmaktadır (Foster and Kesselman, 2004). Örgü sistemlerinin amacı kullanıcılarına sistemin devingenliğini hissettirmeden güçlü ve dayanıklı bir hesaplama ortamı sunmaktır. Örgü sistemleri tarafından kullanıma sunulan kaynakların, kullanıcılar tarafından detaylı teknik bilgiye ihtiyaç duymadan erişilebilmeleri gerekmektedir. Öte yandan görevlerin kaynaklara atanması, yük dengelemesi, kaynak taraması ve kaynak ataması gibi yönetsel görevlerin kullanıcıdan bağımsız olarak arka planda gerçekleştirilmesi gerekmektedir. Örgü sistemleri, koşut ve dağıtık sistemlerden farklı olarak geniş ölçekli, türdeş olmayan ve devingen olarak nitelendirilmektedir (Foster and Kesselman, 2004). Örgü sistemlerde genellikle yüksek miktarlarda kaynak olduğu varsayılmaktadır. Bu kaynaklar hesaplama

kaynakları, veri kaynakları veya hizmet kaynakları gibi farklı kavramlara karşılık gelebilmektedir. Buna karşın, düğümler örgü sistemi içerisindeki bilgisayarlara karşılık gelmektedir. Örgü sistemlerindeki bu düğümler farklı özelliklerde ve devingen olabilmektedir. Düğümlerin devingenlikleri özelliklerinin devingen olmasının yanı sıra örgü sistemlerindeki varlıklarının devingenliğine de karşılık gelebilmektedir. Örgü sisteminde düğümler herhangi bir zamanda haber vermeden sisteme katılıp sistemden ayrılabilir (Foster and Kesselman, 2004). Bütün bu özellikleri ile Örgü sistemleri kullanıcılarına geniş ölçekli kaynaklar sunmaktadırlar. Örgü sistemleri günümüz araştırma ve endüstri alanlarında yaygın olarak kullanılmaktadır. Örneğin, bilişim alanında çalışan araştırmacılar yüksek başarılı hesaplamalar, karmaşık benzetim ortamları, yapay zekâ uygulamaları ve görüntü işleme gibi alanlarda örgü sistemlerini kullanmaktadırlar. Bunun yanı sıra diğer bilimsel araştırma alanlarında çalışan araştırmacılar da karmaşık deneylerde elde edilen verinin yönetimi amacıyla örgü sistemlerini kullanmaktadırlar. Şirketler ve endüstri ise şirketlere özel verinin depolanması ve erişilebilirliği amacıyla örgü sistemlerini kullanmaktadırlar (Foster and Kesselman, 2004).

Örgü sistemlerinin kullanım alanlarının artması ile birlikte arabulucu sistemler gibi mevcut sistemlerin örgü sistemi ortamına taşınması kaçınılmaz hale gelmiştir. Bu aşamada örgü sisteminin dağıtık ve koşut sistemlerden farklı olan özellikleri uyarlamada uyum sorunlarının ortaya çıkmasına sebep olmaktadır. Dağıtık ve koşut sistemlerde geçerli olan varsayımların bazıları örgü sistemleri üzerinde geçerliliklerini yitirmektedir. Geçerliliğini yitiren bu varsayımlar şu şekilde listelenebilmektedir: *i)* düğümler türdeşdir, *ii)* sistem kararlıdır *iii)* sistemde sınırlı sayıda kaynak bulunmaktadır. Bu varsayımların geçersiz olmaları ve örgü sistemlerinin özellikleri, dağıtık ve koşut sistemler için geliştirilmiş olan uygulamaların örgü sistemlerine uyarlanmaları sırasında çeşitli problemlerin ortaya çıkmasına sebep olmaktadır. Dağıtık sorgu işleme, günümüz araştırmalarında, bu konular içerisinde sık irdelenen alanlar arasında bulunmaktadır. Bu problemlerden bazıları, özellikle dağıtık sorgu işleme alanı göz önüne alındığında, şu şekilde listelenebilmektedir: *i)* kaynak taraması, *ii)* kaynak ataması, *iii)* özerk hesaplama ve gözleme servisleri, *iv)* yedekleme ve ön belleğe alma, *v)* güvenlik ve diğer alanlar. Bu problemlerden kaynak tarama ve kaynak atama problemleri günümüz örgü sistemi araştırma konuları arasında sıklıkla irdelenen konular arasında bulunmaktadır. Bu sebeple, bu tezde örgü sistemler üzerinde sorgu işleme için kaynak taraması ve kaynak ataması problemleri üzerinde odaklanılmıştır.

1.2 Örgü Sistemlerde Sorgu İşleme için Kaynak Taraması (KT)

Örgü sistemlerde kaynak taraması, örgü sisteminin özellikleri göz önüne alınarak bir görev için uygun olan kaynakların taranıp bulunması olarak tanımlanmaktadır. Kaynak taraması ortamın devingenlik, türdeş olmama ve geniş ölçeklilik özellikleri dikkate alınarak, güvenilir bir şekilde ve sorgunun işletilmesini iyileştirecek bir sürede gerçekleştirilmesi gerekmektedir. Bu kısıtlar göz önüne alınarak KT problemi için günümüze kadar birçok yöntem önerilmiştir. Mevcut çalışmalar genellikle üç sınıf altında incelenmişlerdir (Hameurlain, et al., 2008; Hameurlain, et al., 2010): merkezi/hiyerarşik sistemler tabanlı KT yöntemleri (Yu, et al., 2003; Foster and Kesselman, 2004; Antonioletti, et al., 2005; Elmroth and Tordsson, 2005; Ramos and Magalhaes, 2006; Kaur and Sengupta, 2007; Molto, et al., 2008), eşler arası sistemler tabanlı KT yöntemleri (Cai, et al., 2003; Cheema, et al., 2005; Marzolla, et al., 2007; Trunfio, et al., 2007), ve etmen sistemleri tabanlı KT yöntemleri (Cao, et al., 2002; Ding, et al., 2005; Kakarontzas and Savvas, 2006; Yan, et al., 2007; Yu, et al., 2006). Her ne kadar önerilen algoritmalar KT problemine zengin çözümler önermiş olsalar da, her KT yöntemi sınıfının kendine has üstünlükleri ve eksiklikleri bulunmaktadır. Detay vermek gerekirse, merkezi/hiyerarşik sistemler tabanlı KT yöntemleri eksiksiz çalışan güvenilir KT yöntemleri olmalarına rağmen örgü sistemlerinin geniş ölçeklilik ve devingenlik özelliklerine uygun değildirler, çünkü dizinleme mekanizmalarının merkezi olması geniş ölçekte dar boğaz problemine, devingen ortamlarda ise tekil hata noktası problemine sebep olmaktadır. Öte yandan, eşler arası sistemler tabanlı KT yöntemleri eşler arası sistemlerin tüm özelliklerinden faydalanmaktadırlar, ancak seçilen eşler arası sistem tipine göre ölçeklenebilirlik, tekil hata noktası veya kaynak taramada kullanılacak özelliklerin kısıtlanması problemleri ile karşılaşabilmektedirler. Son olarak etmen sistemleri tabanlı KT yöntemleri etmenlerin özerklik ve esneklik gibi özelliklerin faydalanırken, etmen sisteminde kullanılacak ilingenin seçimine bağlı olarak darboğaz veya tekil hata noktası problemleri ile karşılaşabilmektedirler. Tüm bu analizler göz önüne alındığında, örgü sistemleri üzerinde sorgu işlemesi için önerilecek olan kaynak tarama algoritmasının şu özellikleri taşıması gerektiği ortaya çıkmaktadır: *i)* ölçeklenebilir olma, *ii)* kaynak taraması sırasında birden çok kaynak özelliğini arayabilme, *iii)* ilingedeki değişikliklere ve düğümlerin sistemden ayrılma durumlarına dayanıklı olma, *iv)* düğüm özellikleri taranırken belirli bir aralıktaki değerleri arayabilme ve *v)* düğümlerin devingen özelliklerini göz önüne alacak şekilde güncel sonuçlar döndürebilme. Bu tez kapsamında hazırladığımız literatür taramasına göre, bilgimiz dahilinde, bahsedilen gereksinimlerin tümünü

karşılayan bir kaynak tarama yöntemi bulunmamaktadır. Bu sebeple, bu tezde ilk olarak bahsi geçen gereksinimlerin tümünü karşılayan bir kaynak tarama algoritması önerilmektedir. Bunu gerçekleyebilmek amacıyla kaynak taraması için kullanılmak üzere verimli ve güvenilir, tüme gönderim ve tümünden toplama işlemlerine uygun bir ilinge denetimi algoritması önerilmektedir. Önerilen ilinge denetimi kapsayan ağaç şeklinde olması planlanmıştır. Örgü sistemlerinin devingenliğine çözüm getirebilmek amacıyla önerilen ilinge denetimi algoritmasının öz-kararlı olması önerilmektedir. İlingenin oluşturulmasından sonra kapsayan ağaç üzerinde kaynak taraması için tüme gönderim ve tümünden toplama yöntemlerini kullanan kapsayan ağaç tabanlı bir kaynak tarama algoritması önerilmektedir. Bunun için ilk olarak hem kaynak tarama hem de öz-kararlı kapsayan ağaç algoritmaları ile ilgili detaylı bir literatür taraması sunulmaktadır. Daha sonra çizgenin merkezi etrafında kök düğümü olan kapsayan ağaç oluşturma amaçlı üç adet öz-kararlı algoritma önerilmektedir. Önerilen algoritmaların karmaşıklık analizleri sunulmakta ve analizler deneylerle güçlendirilmektedir. Önerilen algoritmalar ns2 benzetim ortamında uygulanmakta ve benzer çalışmalar ile kıyaslanmaktadır. Daha sonra, kapsayan ağaç yapısından faydalanan şekilde kapsayan ağaç tabanlı bir kaynak tarama algoritması önerilmektedir. Bu algoritmanın da karmaşıklık analizleri sunulmakta ve analizler ns2 benzetim ortamında gerçekleştirilen deney sonuçları ile güçlendirilmektedir. Sunulan algoritma benzer yöntemlerle karşılaştırılmakta, güçlü ve zayıf yönleri verilmektedir.

1.3 Örgü Sistemlerde Sorgu İşleme için Kaynak Ataması (KA)

Kaynak tarama aşamasının tamamlandığında sorguyu işleyebilecek niteliklerde bir düğüm grubu oluşturulmaktadır. Bu noktadan itibaren, sorgunun içerisindeki görevlerin verimli yürütülebilmeleri için bir kaynak atama algoritmasına ihtiyaç duyulmaktadır. Burada önerilecek olan kaynak atama algoritmasının: (i) kaç adet düğümün sorgu işlemesine katılacağına, (ii) hangi düğümlerin sorgu işlemesine katılacağına ve (iii) sorgu işlemesi sırasında sorguya ait görevleri yürüten düğümlerden bir ya da birkaçının sistemden ayrılması halinde izlenmesi gereken adımlara karar vermesi gerekmektedir. Bu kısıtlar örgü sisteminde sorgu işlemesinin başarımını doğrudan etkilemektedir. Sorgu işlemesi için kaynak ataması, sorgu içerisindeki işletmenlere ait görevleri yürütebilecek en uygun düğümlerin seçilmesi ve atanması olarak tanımlanmaktadır. Örgü sistemleri gibi geniş ölçekli sistemlerde en iyi kaynak atama şeklini bulmak NP-Complete bir problem olarak tanımlanmıştır (Chihping, 1990). Bu sebeple mevcut

alışmalar KA problemine en iyiye yakın yaklaşımsal özmler önermişlerdir. Fakat en iyiye yakın kaynak atama şeklinin bulunması, sorgu işleminin başarımı artırmak için tek başına yeterli olmamaktadır. Aynı zamanda sorgunun işlenmesinde kullanılan düğümlerin sistemden ayrılmaları halinde izlenecek yolun da belirlenmesi gerekmektedir. Örgü sistemlerinin devingen olmaları sebebiyle bu gibi hata durumları kaçınılmaz olmaktadır. Bu türden hata durumları özellikle uzun sürecek sorguların işletilmesi sırasında çok masraflı olabilmektedir. Bu sebeple örgü sistemleri üzerinde sorgu işleminde kullanılacak olan kaynak atama algoritmasının hataya dayanıklı olarak tasarlanması büyük önem taşımaktadır. Literatürde örgü sistemleri üzerinde sorgu işlemesi için bir çok kaynak atama algoritması bulunabilmektedir (Gounaris, et al., 2004; Gounaris, et al., 2005; Soe, et al., 2005; Gounaris, et al., 2006; Silva, et al., 2006; Bose, et al., 2007; Kotowski, et al., 2008; Liu and Karimi, 2008; Gounaris, et al., 2009). Bu algoritmalar birçok alışmada sınıflandırılarak değerdendirilmiştir (Costa and Furtado, 2008; Epimakhov, et al., 2011; Cokuslu, et al., 2012a). Her ne kadar mevcut KA alışmaları ilgi çekici özmler sunmuş olsalar da, bilğimiz dâhilinde aday düğüm arama uzayının ölçeğini daraltan, ölçeklenebilir alışmalar bulunmamaktadır. Buna ek olarak sorgu işlemesi sırasında oluşacak iletişim maliyetlerini azaltmaya yönelik çok az alışma bulunmaktadır (Bose, et al., 2007; Liu and Karimi, 2008). Bahsi geen mevcut alışmaların birçoğunda aday düğümlerin özelliklerinin devingenlikleri irdelenmiştir. Fakat örgü sisteminin ve ilingenin devingenliğı çok az alışmada irdelenmiştir (Smith and Watson, 2005; Smith and Watson, 2007; Taylor, 2008; Bestehorn, et al., 2010). Bu alışmaların hiçbirisi kodlayarak birleştirme gibi konumlu işletmenlerin alıştırılması üzerine odaklanmamıştır. Mevcut literatürdeki analizlerimiz göz önüne alındığında bu alanda hâlâ katkı konulabilecek açık konuların bulunduğuna inanmaktayız. Bu sebeplerden dolayı, bu tezde kaynak tarama algoritmalarının yanı sıra örgü sistemlerde sorgu işlemesi için kaynak atama algoritmaları da önerilmektedir. Bunun için öncelikle kaynak atama algoritmaları ve hataya dayanıklı sorgu işlemeli bir literatür taraması alışması sunulmaktadır. Sonra, tek bir birleştirme işletmeni için kaynak ataması geekleştiren, aday düğümlerin arama uzayını daraltıp veri kaynaklarına uzaklıklarını gözetererek kaynak ataması geekleştiren yeni bir kaynak atama algoritması önerilmektedir. Daha sonra, birden çok birleştirme işletmeninden oluşan bir sorgunun bütününe kaynak ataması geekleştiren bir kaynak atama algoritması önerilmektedir. Son olarak sorgu işleminin hataya dayanıklı hale getiren hataya dayanıklı bir kaynak atama algoritması önerilmektedir. Önerilen algoritmalar için karmaşıklık analizleri sunulmakta ve bu analizler benzetim ortamında geekleştirilen testler ile

doğrulanmaktadır. Algoritmalar benzer mevcut algoritmalar ile kıyaslanmakta, üstünlükleri ve eksiklikleri sunulmaktadır.

1.4 Deneysel Doğrulama

Örgü sistemleri bilgi teknolojileri alanında göreceli olarak yeni bir kavramdır. Her ne kadar literatürde birçok ağ benzetim ortamı bulunsa da hemen hemen hiçbirisi örgü sistemlerini eksiksiz olarak modellememektedir. Bu sebeple, tezimizde önerilen algoritmaların test edilmesinde kullanılabilecek benzetim ortamları araştırılmıştır. Bu kapsamda, GridSim, OptorSim, OPNET ve ns2 benzetim ortamları amacımıza uygun olup olmamalarına göre incelenmiş, GridSim ve ns2 olmak üzere iki adet benzetim ortamının kullanılmasına karar verilmiştir.

GridSim (Sulistio, et al., 2008), örgü sistemi üzerinde görev planlama yöntemlerinin geliştirilebilmesi amacıyla hazırlanmış bir ayrık olay benzetimcisidir (*ing. discrete event simulator*). GridSim’de kaynaklar işlemci hızları (MIPS), işlemci sayısı, işlemci yükü ve bağlantı hızı (baud hızı) dikkate alınarak modellenmiştir. Görevler ise girdi verisi büyüklüğü, yönerge sayısı ve çıktı verisi büyüklüğü dikkate alınarak ifade edilmiştir. OptorSim (Bell, et al., 2003) bir diğer örgü benzetim ortamıdır. EU Data Grid (Gagliardi, et al., 2002) örgü ortamını benzetmek için tasarlanmıştır. Benzetim, örgünün hesaplama ve veri depolama hizmetlerini sunan birçok kümeden oluştuğu varsayılarak yapılmaktadır. Hesaplama Birimleri (HB), Depolama Birimlerinde (DB) depolanmış veriyi kullanan görevleri yürütmekte ve tek bir kaynak yöneticisi tarafından yönetilmektedir. OptorSim’de kaynaklar her bir hesaplama kümesi içerisindeki HB sayısı, DB sayısı ve büyüklüğü ve kümeler arasındaki bant genişliği değerleri ile temsil edilmektedir. Görevler ise dosya gereksinimleri, kullanılacak dosyaların büyüklükleri ve ihtiyaç duyulan HB özellikleri ile temsil edilmektedir. Bu benzetim ortamlarına farklı bir seçenek de OPNET’tir. Diğerlerinden farklı olarak OPNET ticari bir ürün halinde bulunmaktadır. Birçok farklı ağ ortamları oluşturmaya müsait güçlü bir benzetim ortamıdır. OPNET benzetim ortamı oluşturulabilmesi için birçok farklı araç sunmaktadır. Dağıtık hesaplama deneylerinin yapılması için örgü sistemi benzetimini desteklemektedir. Uygulama seviyesinde kullanıcıların kendi hazırladıkları uygulamaları deneyebilmelerine olanak sunmaktadır. Ns2 (Network Simulator 2, version 2.34) (Fall and Varadhan, 2010), açık kaynak bir ayrık olay benzetim ortamı sunmaktadır. Ns2, TCP ve UDP protokollerini, kablolu ve kablosuz ağ

ortamlarında çeşitli yönlendirme protokolleri gibi birçok farklı katmanda çeşitli protokolleri benzetmektedir. Ns2, ilk olarak 1989'da REAL ağ benzetim ortamının geliştirilmiş hali olarak kullanıma sunulmuştur. Yıllar içerisinde UCB Daedalus, CMU Monarch projeleri ve Sun Microsystems gibi diğer araştırmacıların da katkılarıyla genişletilmiş ve günümüzdeki halini almıştır. Ns2, açık kaynak olması ve geniş kapsamlı bir benzetim ortamı olması sebebiyle günümüzde araştırmacılar tarafından çokça tercih edilen bir benzetim ortamı haline gelmiştir.

Her ne kadar bu benzetim ortamları örgü sistemleri için geliştirilen uygulamaların test edilmesi için güçlü benzetim ortamları sunsalar da her birinin kendine göre zayıf ve güçlü yanları bulunmaktadır. Örneğin, OPNET kapsamlı bir benzetim ortamı sunmasına karşın açık kaynak olmaması sebebiyle bu tezde kullanılamamıştır. Buna benzer olarak OptorSim örgü ortamının benzetimini sunmakla beraber, kullanıcı uygulamalarının benzetim ortamında test edilmesine olanak sunmamaktadır. Bu özellikleri ile mevcut benzetim ortamları irdelendiğinde, bu tezde önerilen algoritmaların test edilmesi amacıyla GridSim ve ns2 benzetim ortamlarının kullanılmasına karar verilmiştir.

1.5 Katkılar

Bu tezde örgü sistemlerde sorgu işlemesi için kaynak tarama ve kaynak atama algoritmaları önerilmektedir. Örgü sistemlerinin ayırt edici özellikleri belirlenmiş ve kaynak tarama ve kaynak atama algoritmaları açısından bu özelliklerden kaynaklanan problemlere çözüm bulmak amacıyla algoritmalar önerilmektedir. Kaynak tarama alanı için bu tezde sunulan katkı aşağıdaki şekilde listelenmektedir:

i. Öz-kararlı kapsayan ağaç oluşturma algoritmaları: Bu alanda, 3 adet yeni öz-kararlı kapsayan ağaç oluşturma algoritması önerilmektedir. Bu algoritmalar ile küçük yarıçaplı kapsayan ağaçlar oluşturarak ölçeklenebilirlik problemine çözüm önerilmektedir. Önerilen algoritmaların öz-kararlı olacak şekilde tasarlanmasıyla da örgü sistemlerinin devingenliğinden kaynaklanan problemlere çözüm önerilmektedir. Önerilen algoritmalar benzer çalışmalarla karşılaştırılarak getirilen katkı gösterilmektedir.

ii. Kapsayan ağaç tabanlı kaynak tarama algoritması: Kapsayan ağacın özelliklerinden faydalanarak, güncel kaynak bilgisinin erişilmesini

sağlayan verimli bir kaynak tarama algoritması önerilmektedir. Bu algoritma ile her düğümün özellikleri keşfedilerek örgü sistemindeki düğümlerin türdeş olmamasından kaynaklanan problemlere çözüm önerilmektedir. Aynı zamanda kapsayan ağaç üzerinde tüme gönderim ve tümünden toplama yöntemlerinin kullanılması ile ölçeklenebilirlik problemine de çözüm önerilmektedir.

Kaynak Atama problemi kapsamında ise, tezde sunulan katkı aşağıda listelenmektedir:

i. Tekli Birleştirme İşletmeni Kaynak Atama (TBİKA) Algoritması: TBİKA algoritmasında, tek bir birleştirme işletmeninden oluşan sorgulara kaynak ataması gerçekleştiren bir kaynak atama algoritması önerilmektedir. TBİKA algoritması aşağıda listelenen, sıralı çalışacak iki algorithmadan oluşmaktadır:

Yakınlık Tabanlı Aday Düğüm Listesi Oluşturma (YADLO) Algoritması: Bu algorithmada birleştirme işletmeni için kullanılabilecek aday düğüm listesinin arama uzayı kısıtlanmaktadır. Bu kısıtlama kaynak atama algoritmasının ölçeklenebilirliğini artırmaktadır. Aday listesi oluşturulurken veri kaynaklarına yakın olan adayların listeye eklenmesi sağlanmakta ve listedeki aday düğümler veri kaynaklarına olan yakınlıklarına göre sıralanmaktadır. Bu algoritma ile sorgu işlemesi sırasında oluşacak iletişim maliyetlerinin azaltılması amaçlanmaktadır.

Birleştirme Görevi Kaynak Atama (BGKA) Algoritması: Bu algoritma YADLO ile oluşturulan aday düğüm listesinin içerisinde en uygun aday düğümlerin seçilmesini sağlamaktadır. Bu algoritma ile örgü sistemindeki düğümlerin türdeş olmamasından kaynaklanan problemlere çözüm önerilmektedir.

ii. Çoklu Birleştirme İşletmeni Kaynak Atama (ÇBİKA) Algoritması: Bu algoritma ile birden fazla birleştirme işletmeninden oluşan sorgulara kaynak ataması gerçekleştirecek bir algoritma önerilmektedir. ÇBİKA algoritması ile, TBİKA algoritmasından faydalanarak ölçeklenebilirlik ve türdeş olmayan kaynaklar problemlerine çözüm önerilmektedir.

iii. Hataya Dayanıklı Kaynak Atama (HDKA) Algoritması: HDKA algoritması ile sorgunun işletilmesi sırasında hataya dayanıklılık

sağlanmaktadır. Bu şekilde örgü sistemlerinin devingenliğinden kaynaklanan problemlere çözüm önerilmektedir.

1.6 Tezin Yapısı

Bu dokümanın geri kalanı aşağıdaki şekilde yapılandırılmıştır: Bölüm 2’de örgü sistemlerinde sorgu işleme için mevcut kaynak taraması ve ataması yöntemleri sınıflandırılarak detaylı olarak incelenmektedir. Bu konudaki mevcut çalışmalar incelenmekte ve her birinin güçlülükleri ve zayıflıkları sınıflar arası karşılaştırma ile sunulmaktadır. Kaynak tarama ve kaynak atama alanındaki açık noktalar ve olası katkılar listelenmektedir. Bölüm 3’te kaynak tarama problemi üzerine odaklanılmaktadır. Öncelikle kaynak taramasında kullanılacak verimli bir ilinge oluşturulmak amacıyla öz-kararlı kapsayan ağaç oluşturma algoritmaları önerilmektedir. Daha sonra kapsayan ağaç tabanlı kaynak tarama algoritması önerilmektedir. Bölüm 4’te örgü sistemlerinde sorgu işlemesi için başlangıç kaynak ataması gerçekleştirecek algoritmalar önerilmektedir. Bu amaçla öncelikle tekli birleştirme işletmeni kaynak atama algoritması önerilmektedir. Daha sonra birden fazla birleştirme işletmeninden oluşan sorgulara kaynak ataması gerçekleştirecek olan çoklu birleştirme işletmeni kaynak atama algoritması önerilmektedir. Bölüm 5’te hataya dayanıklı kaynak atama algoritması önerilmektedir. Son olarak, bölüm 6’da sonuçlar ve gelecek çalışmalar sunulmaktadır.

2. LİTERATÜR ÖZETİ

Bu bölümde örgü sistemlerde sorgu işleme için kaynak taraması ve kaynak ataması konuları üzerine detaylı bir literatür taraması sunulmaktadır. Öncelikle kaynak taraması alanında sunulmuş çalışmalar sınıflandırılarak incelenmektedir. Mevcut kaynak taraması çalışmaları hakkında sunulan değerlendirmenin ardından kaynak taramasında ilinge denetimi amacıyla kullanılacak öz-kararlı kapsayan ağaç oluşturma algoritmaları incelenmektedir. Daha sonra kaynak ataması alanında sunulmuş mevcut çalışmalar sınıflandırılarak incelenmektedir. Son olarak ta hataya dayanıklı kaynak atama çalışmaları incelenmektedir.

2.1 Giriş

Örgü sistemleri geniş ölçekli hesaplama kabiliyetleri ve yüksek erişilebilirlikleri sebebiyle günümüzün ilgi çeken araştırma konularından biri haline gelmiştir. Örgü sistemlerin geniş hesaplama ve depolama kapasiteleri birçok araştırmacının dikkatini çekmekte ve mevcut sistemlerini örgü sistemleri üzerinde çalışacak şekilde uyarlamaya yönlendirmektedir. Dağıtık sorgu işleme de üzerinde birçok çalışmanın yapıldığı alanlardan birisidir (Huang, et al., 2003; Antonioletti, et al., 2005; Pacitti, et al., 2007; Kotowski, et al., 2008; Taniar, et al., 2008). Örgü sistemlerinin türdeş olmayan kaynaklardan oluşma, devingenlik ve geniş ölçeklilik özellikleri, dağıtık sorgu işleminin örgü sistemlerine uyarlanması esnasında birçok yeni araştırma konusunun ortaya çıkmasına sebep olmaktadır. Bu problemlerden öne çıkanları kaynak taraması, kaynak ataması, kaynak seçimi, veri yönetimi, süreç gözlemlene, güvenlik konuları ve benzer konular olarak gösterilmektedir (Gounaris, et al., 2005). Bu tezde örgü sistemleri üzerinde sorgu işleme için kaynak taraması ve kaynak ataması konuları üzerine odaklanılmaktadır. Bu konudaki problemleri daha iyi anlayabilmek amacıyla, bu bölümde kaynak taraması ve kaynak ataması konuları alanındaki mevcut çalışmalar incelenerek detaylı bir literatür taraması sunulmaktadır. İlk olarak, kısım 2.2’de, mevcut kaynak tarama yöntemleri sınıflandırılarak incelenmektedir. Daha sonra, kaynak tarama yöntemleri hakkındaki değerlendirmelere dayanarak, kaynak taramasında kullanılmaya müsait ilinge denetimi algoritmaları ile ilgili bir literatür özeti sunulmaktadır. Bunun için, örgü sistemlerinin özelliklerinden kaynaklanan devingenlik ve ölçeklenebilirlik problemlerine çözüm olacak öz-kararlı kapsayan ağaç algoritmaları incelenmektedir. Kısım 2.3’te örgü sistemleri üzerinde sorgu işleme için kaynak ataması yöntemleri ile ilgili detaylı bir literatür taraması sunulmaktadır. Daha sonra, örgü sistemlerinin devingenliği düşünülerek,

hataya dayanıklı kaynak atama algoritmaları ile ilgili bir literatür taraması sunulmaktadır. Son olarak, kısım 2.4'te, mevcut çalışmalar ile ilgili sunulan literatür çalışması göz önüne alınarak varılan sonuçlar ve bu alandaki açık ve katkı konulmaya müsait noktalar sunulmaktadır.

2.2 Örgü Sistemlerinde Sorgu İşleme İçin Kaynak Taraması (KT)

Örgü üzerinde kaynak taraması (KT), işlem ihtiyaçlarının ve ortam değişkenlerinin açıkça belirtildiği bir ortamda işlemi yürütebilme yetisine sahip olan kaynakların örgü içerisinde araştırılması olarak tanımlanmaktadır. Bununla birlikte KT geniş ölçekli, devingen ve karışık özelliklere sahip bir ortamda kabul edilebilir bir süre içerisinde tamamlanmalıdır. Örgü sistemlerinin başarılı biçimde kullanılması uygun kaynakların bulunmasına bağlıdır. Bu sebeple KT uygun kaynakların bulunmasında önemli bir adım olarak kabul edilmektedir. Ancak örgü sistemlerinin özellikleri KT işleminin zaman alan bir süreç haline gelmesine sebep olmaktadır. Örgü üzerinde KT problemini çözmek için birçok çalışma yapılmıştır. Bu çalışmalar genel olarak üç sınıfta incelenmektedir (Cokuslu, et al., 2009; Cokuslu, et al., 2010b; Hameurlain, et al., 2008; Hameurlain, et al., 2010): (i) merkezi/sıradüzensel yöntemler, (ii) eşler arası (P2P) tabanlı yöntemler ve (iii) etmen tabanlı yöntemler.

Bu kısımda örgü sistemlerde sorgu işleme için kaynak taraması yöntemleri hakkında bir literatür taraması sunulmaktadır. Bahsedilen kaynak taraması yöntemleri üç sınıf altında incelenmiştir: (i) merkezi/sıradüzensel KT yöntemleri, (ii) etmen tabanlı KT yöntemleri ve (iii) eşler arası (P2P) sistemler tabanlı KT yöntemleri. KT yöntemleri incelenirken bu tezde kullanılan sınıflandırma şekil 2.1'de gösterilmektedir. Her bir sınıf için yöntemler incelenmiş, ardından detaylı çözümlemeler verilmiş ve sınıflar arası karşılaştırmalar yapılmıştır. Karşılaştırmalar karmaşıklık, ölçeklenebilirlik, devingenlik, güvenilirlik, çoklu öznitelik desteği, devinimli öznitelik desteği ve aralıklı öznitelik desteği ölçütlerine göre gerçekleştirilmiştir. Bu ölçütlerin örgü uygulamalarında önemi aşağıda listelenmektedir:

- Karmaşıklık, yöntemlerin performansları ile ilgili sayısal değerler vermektedir. Bu tezde karmaşıklık zaman ve mesaj karmaşıklığı olmak üzere iki alanda incelenmektedir. Mesaj karmaşıklığı algoritmanın amacına ulaşması için sistemde gönderilen toplam mesaj sayısını göstermektedir. Mesaj karmaşıklığının yüksek olması algoritmanın

başarımının düşük olmasına sebep olabilmektedir. Öte yandan zaman karmaşıklığı algoritmanın amacına ulaşmak için kaç adıma ihtiyacı olduğunu göstermektedir.

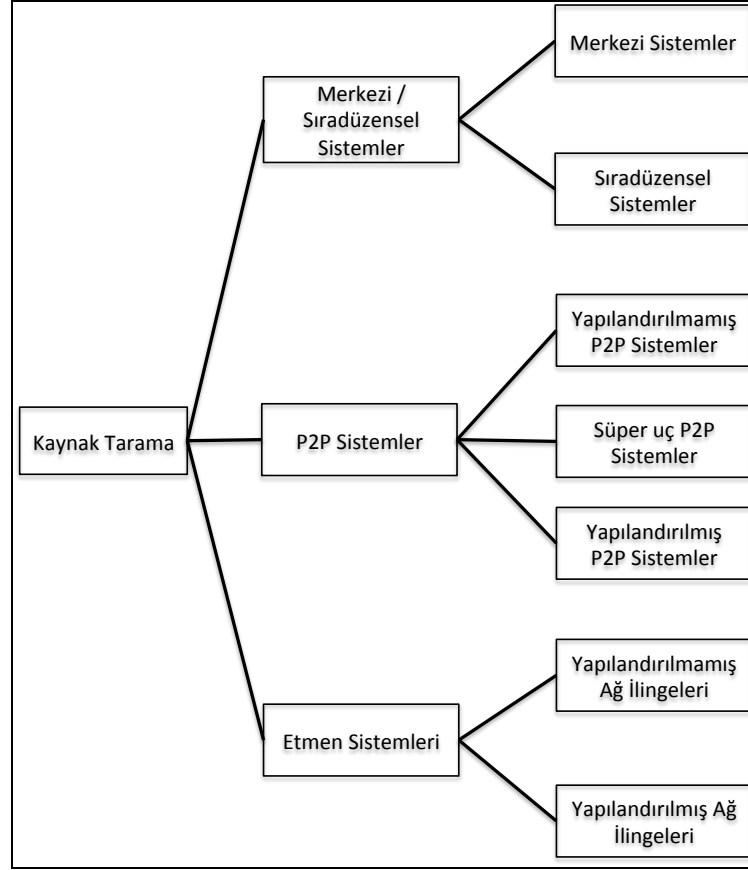
- Ölçeklenebilirlik, örgü sistemlerinin geniş ölçekli sistemler olması sebebiyle önem taşımaktadır. Ölçeklenebilir olmayan bir algoritmanın başarımı, çalışılan ortamın boyutlarının büyümesiyle hızla düşmektedir. Bu durum önerilen algoritmanın örgü sistemine olan uygunluğunun bir göstergesi olarak kullanılabilir.

- Devingenlik, yaklaşımların devingen bir ortamda nasıl davranacağı konusunda bilgi vermesi ve örgü sistemlerinin devingen olması açısından önem taşımaktadır. Örgü sistemleri devingen ortamlar olmaları sebebiyle devingenliğe dayanlı algoritmalar gerektirmektedir.

- Güvenilirlik ise örgü üzerinde çalışacak kimi uygulamaların hatalı sonuçlara toleransı olmaması sebebiyle önem taşımaktadır. Örgü sisteminde aranan bir düğümün var olmasına rağmen olumsuz sonuç dönülmesi veya sistemde olmamasına rağmen olumlu sonuç dönülmesi algoritmanın güvenilir olmadığını göstermektedir.

- Çoklu özellik tarama, devingen özellik tarama ve aralıksal özellik tarama da aranacak kaynakların farklı tipte özelliklerinin taranabilmesini göstermektedir. Çoklu özellik tarama, aranacak kaynakların birden fazla arama ölçütü ile taranabilmesini sağlamaktadır. Devingen özellik tarama da aranan düğümlerin devingen özelliklerinin anlık olarak güncel haliyle taranabilmesini göstermektedir. Aralıksal özellik tarama ise aranacak sayısal düğüm özelliklerinin kesin değerler yerine belirli bir aralığa düşen değerler arasından aranabilmesini göstermektedir.

Bu kısımda ayrıca, kaynak taramasında ilinge denetiminde kullanılmak üzere, öz-kararlı kapsayan ağaç oluşturma algoritmaları hakkında da bir literatür özeti sunulmaktadır.



Şekil 2.1 Örgü sistemlerinde sorgu işleme için kaynak taraması algoritmaları sınıflandırması

2.2.1 Merkezi ve sıradüzensel sistemler tabanlı örgü kaynak taraması

Merkezi ve sıradüzensel sistemler örgü servisleri için uygun ve kolay erişilebilir araçlar sunmaktadırlar (Antonioletti, et al., 2005; Elmroth and Tordsson, 2005; Kaur and Sengupta, 2007; Molto, et al., 2008; Ramos and Magalhaes, 2006; Yu, et al., 2003). Bu tür sistemlerde kaynak bilgisi merkezi veya sıradüzensel sunucularda tutulmaktadır ve kaynak taraması bu sunucuların sorgulanması ile gerçekleştirilmektedir. Merkezi ve sıradüzensel kaynak tarama yöntemleri çoğunlukla web servisleri tarafından kullanılmaktadır (Antonioletti, et al., 2005). Bu kısımda merkezi ve sıradüzensel sistemler tabanlı kaynak tarama yöntemleri ile ilgili detaylı bir literatür taraması sunulmaktadır (Beheshti and Moshkenani, 2007; Czajkowski, et al., 2001; Elmroth and Tordsson, 2005; Fitzgerald, et al., 1997; Foster and Kesselman, 1997; Kaur and Sengupta, 2007; Li, et al., 2002; Molto, et al., 2008; Ramos and Magalhaes, 2006; Riedel, et al., 2007; Yu, et al., 2003). Bu çalışmalar kaynak bilgisinin saklandığı ilingenin

şekline göre sınıflandırılmıştır. Merkezi yöntemlerde kaynak bilgisi merkezi sunucularda, sıradüzensel yöntemlerde ise kaynak bilgisi kısmi olarak bölünerek birden çok sunucu üzerinde saklanmaktadır. Bu yöntemlerin her biri için özet analizler verilmekte ve yukarıda tanımlanan değerlendirme ölçütlerine göre değerlendirilmeleri sunulmaktadır.

2.2.1.1 Merkezi sistemler tabanlı örgü kaynak taraması

Örgü ortamlarında merkezi sistemler kullanılarak kaynak taraması üzerinde çokça çalışılmış bir konudur. Bu sınıfa dâhil olan çalışmalarda ana fikir örgü sistemindeki kaynaklara ait bilginin merkezi sunucularda tutulması esasına dayanmaktadır. Bu sınıftaki çalışmaların çoğu LDAP (Lightweight Directory Access Protocol) veya UDDI (Universal Description Discovery and Integration) gibi mevcut kaynak bilgisi yönetim sistemlerini kullanmaktadır. Bu sınıftaki çalışmaların çoğu, örgü sistemlerinin ilk zamanlarında sıklıkla kullanılmışlardır (Fitzgerald, et al., 1997; Kaur and Sengupta, 2007; Molto, et al., 2008; Riedel, et al., 2007; Yu, et al., 2003). Örneğin, Yu ve arkadaşları (Yu, et al., 2003) web servisleri tabanlı bir örgü servis yayınlama sistemi (GMD) geliştirmişlerdir. Bu sistem, kullanıcılar ve kaynaklar için, XML biçimli mesajlar kullanarak, web üzerinden servis sağlamaktadır. Kaur ve Sengupta, örgü sistemleri için web servisleri tabanlı bir kaynak tarama yöntemi geliştirmiştir (Kaur and Sengupta, 2007). Molto ve arkadaşları (Molto, et al., 2008) örgü sisteminde kaynak atamak ve süreç yürütmek isteyen kullanıcıların ağ üzerinden kullanabileceği bir sistem geliştirmişlerdir. Önerilen sistem web üzerinden herhangi bir düzlem gereksinimi olmadan kullanılabilen servis odaklı bir mimari sunmaktadır. Riedel ve arkadaşları (Riedel, et al., 2007), çalışmalarında kaynakların web servisleri tarafından temsil edildiği bir örgü yönetim sistemi sunmuşlardır. (Fitzgerald, et al., 1997) çalışmasında yazarlar merkezi LDAP sunucuları kullanan bir örgü kaynak yönetimi sistemi sunmuşlardır.

Merkezi sistemler tabanlı örgü kaynak tarama yöntemleri örgü kaynaklarının yönetimi için kullanışlı bir ara yüz sunmaktadır. Örgü sistemi içerisindeki kaynakların bilgilerini merkezi veritabanlarında tutmaktadırlar. Kaynak tarama servisinin merkezi olarak saklanması, geniş ölçekli örgü sistemlerinde kolayca darboğaz ve tekil hata noktası problemlerine sebep olabilmektedir. Darboğaz problemi hem sıklıkla gerçekleşen kaynak bilgisi güncellemeleri hem de büyük sayıdaki sorgulamalar sebebiyle gerçekleşebilir. Merkezi algoritmalar devingen örgü sistemlerinde tekil hata noktası problemlerine

de sebep olmaktadır. Kaynak taraması görevini yürüten merkezi bir düğümün aksaması bütün sistemin erişilemez hale gelmesine sebep olabilmektedir. Bazı yaklaşımlarda merkezi düğümlerin çoğaltılması fikri uygulanmış olsa da bu düğümlerin çoğaltılmasından kaynaklanan maliyet geniş ölçekli sistemlerde çok yüksek olabilmektedir. Bu alanda geliştirilen sistemlerin büyük çoğunluğu devinimli öznitelik sorgularını desteklememelerine rağmen çoklu öznitelik ve aralıklı öznitelik sorgularını desteklemektedirler.

2.2.1.2 Sıradüzensel sistemler tabanlı örgü kaynak taraması

Örgü sistemlerinin ölçeğinin genişlemesi ile birlikte merkezi sistemlerin karşılaştığı sorunlara çözüm aramak amacıyla araştırmacılar dikkatlerini sıradüzensel sistemlere doğrultmuşlardır. Bunun bir sonucu olarak sıradüzensel sistemler tabanlı birçok kaynak tarama yöntemi önerilmiştir (Beheshti and Moshkenani, 2007; Elmroth and Tordsson, 2005; Li, et al., 2002; Ramos and Magalhaes, 2006; Yin, et al., 2007). Elmroth ve Tordsson (Elmroth and Tordsson, 2005) çalışmasında web servisleri tabanlı bir örgü kaynak aracı sistemi ve görev gönderi sistemi geliştirmişlerdir. Tasarımlarında, mevcut olan sistemlerden birçoğunu kullanarak ve geliştirerek fayda sağlamışlardır. Kaynak tarama aşaması için kaynaklar hakkında sıradüzensel olarak bilgi sağlama özelliğine sahip olan *Grid Laboratory Uniform Environment (GLUE)* projesinden faydalanmışlardır. Ramos ve arkadaşları (Ramos and Magalhaes, 2006) çalışmasında kaynak taraması için Globus Toolkit (GT3) tabanlı bir web servisi geliştirmişlerdir. Çalışmalarında, örgünün sanal yapılara (SY) ayrıldığı sıradüzensel bir ilinge oluşturmuşlardır. Her bir SY’de yönetici ve çalışan düğümler bulunmaktadır. Yönetici düğümler kaynak bilgilerinin tutulduğu veritabanlarının güncellenmesinden sorumlu iken çalışan düğümler kaynak bilgilerinin toplanıp yönetici düğüme iletilmesinden sorumludur. Beheshti ve Moshkenani etmenleri, ontolojileri ve web servislerini birlikte kullanarak bir kaynak tarama yöntemi sunmuşlardır (Beheshti and Moshkenani, 2007). Yazarlar çalışmalarında kaynak bilgilerinin ayrıcalıklı düğümlere dağıtıldığı bir yöntem kullanmışlardır. (Li, et al., 2002) çalışmasında Li ve arkadaşları ağ paket yönlendirme protokollerinden esinlenerek bir örgü kaynak tarama yöntemi sunmuşlardır. Yin ve arkadaşları (Yin, et al., 2007) çalışmasında 3 aşamalı bir sıradüzensel kaynak tarama yöntemi önermişlerdir.

Sıradüzensel sistemler tabanlı örgü kaynak tarama yöntemleri, merkezi yöntemlere göre benzer faydaları sağlarken aynı zamanda daha ölçeklenebilir bir

düzlem ortaya koymaktadırlar. Geniş ölçekli örgü ortamlarında servislerin sıradüzensel bir ilinge oluşturması darboğaz probleminin oluşma olasılığını düşürmektedir. Fakat sıradüzensel yapının yönetiminde görev alan düğümlerin hatalı çalışması durumunda örgü üzerinde birçok kaynağın erişilemez duruma gelmesi bu sistemlerde tekil hata noktası problemini ortaya çıkartmaktadır. Bu sınıf içerisinde geliştirilen yöntemlerin tamamı çoklu öznitelik ve aralıklı öznitelik sorgularını desteklemektedir. Devinimli öznitelik sorguları ise kaynak taramasından sonra doğrulama yapılması halinde desteklenmektedir.

2.2.1.3 Karşılaştırma

Merkezi ve sıradüzensel kaynak tarama yöntemlerinin sınıflar arası karşılaştırmaları çizelge 2.1’de gösterilmektedir.

Çizelge 2.1 Merkezi ve sıradüzensel kaynak tarama yöntemleri sınıfları arasında karşılaştırma

	Merkezi Sistemler Tabanlı KT	Sıradüzensel Sistemler Tabanlı KT
Ölçeklenebilirlik	Darboğaz problemi sebebiyle ölçeklenebilir değil	Sıradüzensellik sebebiyle daha fazla ölçeklenebilir
Devingenlik	Düğümmlerin devingenliğine dayanıklı, yönetici düğüm devingenliğine dayanıklı değil	Düğümmlerin devingenliğine dayanıklı, yönetici düğüm devingenliğine dayanıklı değil
Güvenilirlik	Kaynak tarama sonuçları anlamında güvenilir, fakat tekil hata noktasına sahip	Kaynak tarama sonuçları anlamında güvenilir, fakat tekil hata noktasına sahip
Aralıklı Özellikleri Tarayabilme	Merkezi kaynak yönetimi sayesinde destekleniyor	Sıradüzensel kaynak yönetimi sayesinde destekleniyor
Çoklu Özellikleri Tarayabilme	Merkezi kaynak yönetimi sayesinde destekleniyor	Merkezi kaynak yönetimi sayesinde destekleniyor
Devingen Özellikleri Tarayabilme	Aralıklı kaynak bilgisi güncellemeleri sebebiyle desteklenmiyor	Aralıklı kaynak bilgisi güncellemeleri sebebiyle desteklenmiyor

Merkezi ve sıradüzensel olan her iki örgü kaynak taraması yaklaşımı da kıyaslama ölçütleri göz önüne alındığında birbirleri ile benzerlik

göstermektedirler. Aralarındaki en belirgin fark ölçeklenebilirlik ve güvenilirlik ölçütlerinde görülmüştür. Merkezi sistemler geniş ölçekli örgülerde darboğaz problemiyle karşılaşmaktadırlar. Bu sistemler için aynı zamanda tekil hata noktası problemi de bulunmaktadır. Her ne kadar kimi çalışmalarda merkezi sunucuların çoklanması ile bu problem ortadan kaldırılmaya çalışılsa da geniş ölçekli örgü sistemlerinde çoklanma işleminin maliyeti göz önüne alındığında bu çözümün uygulanabilirliği düşüktür. Buna karşılık, sıradüzensel sistemler merkezi sunucuların tersine, kaynak taraması yükünü birçok noktaya dağıtmaktadırlar. Bu özellik sistemin ölçeklenebilirliğini arttırmakta, tekil hata noktası probleminin zararlı etkilerini ise azaltmaktadır. Bu sistemlerde bir dizin sunucusunun hatalı çalışması, merkezi sistemlerin aksine, sistemin tamamı yerine daha küçük bir parçasının etkilenmesine sebep olmaktadır. Devingen öznitelik sorguları, sorgunun kaynak düğümlerde çözülmesini gerektirmektedir. Sıradüzenli sistemlerdeki birçok çalışmada sorguların yönetici düğümlerde çözülmesi sebebiyle devingen sorgular desteklenememektedir. Buna rağmen, çoklu öznitelik ve aralıklı öznitelik sorguları incelenen çalışmaların tümünde desteklenmektedir.

2.2.1.4 Sonuçlar

Bu kısımda merkezi ve sıradüzensel örgü kaynak taraması yöntemleri ile ilgili detaylı bir literatür taraması sunulmuştur. İncelenen çalışmalar belirlenen değerlendirme ölçütlerine göre değerlendirilmiş ve sınıflar arası karşılaştırmalar sunulmuştur. Değerlendirmeler göz önüne alındığında merkezi yaklaşımların geniş ölçekli örgü sistemleri için uygun olmadığı söylenebilir. Buna karşı, bu sistemler ölçeğin küçük olduğu ve dizin sunucularının güvenilir olduğu ortamlarda verimli olarak kullanılabilir. Öte yandan, sıradüzensel yöntemler ölçeğin nispeten daha geniş olduğu ortamlara daha iyi uyum sağlayabilmektedir. Fakat yine de ölçeğin ve devingenliğin fazla olduğu örgü sistemlerinde darboğaz ve tekil hata noktası problemleri bu sistemlerin kullanımlarına engel olabilmektedir.

2.2.2 Eşler arası sistemler (P2P) tabanlı örgü kaynak taraması

Örgü sistemlerinin ilk zamanlarında merkezi ve sıradüzensel sistemler tabanlı örgü kaynak tarama yöntemleri sıkça kullanılmaktaydı. Fakat örgü sistemlerinin ölçeklerinin büyümesiyle birlikte merkezi ve sıradüzensel sistemler tabanlı kaynak tarama yöntemleri yetersiz kalmışlardır. Bu sebeple Eşler Arası (P2P) sistemler günümüz bilgisayar dünyasında büyük bir kullanım alanına

sahiptirler. Örgü sistemleri ve eşler arası sistemler arasındaki etkileşim birçok araştırmacı tarafından irdelenmiştir (Iamnitchi and Foster, 2003; Iamnitchi and Talia, 2005; Talia and Trunfio, 2005). P2P sistemleri, geniş ölçekli ortamlarda verimli çalıştığı kanıtlanmış sistemlerdir. Bu konuda sunulmuş detaylı bir literatür taraması (Androutsellis-theotokis and Spinellis, 2004) çalışmasında sunulmuştur. Bu sistemlere olan ilgi sebebiyle zaman içerisinde P2P sistemler gelişmiş ve farklı türlerde yapılanmışlardır. Bu farklı türler genel olarak üç sınıfta incelenmektedir: (i) yapılandırılmamış P2P, (ii) süper uç ve (iii) yapılandırılmış P2P sistemler (Hameurlain, et al., 2010). Bu sistemlerin dağıtık ve geniş ölçekli olma özellikleri, araştırmacıların, örgü kaynak tarama konusunda P2P sistemlerinin uygun araçlar olarak kullanılabileceğini düşünmelerine sebep olmuştur. Bu alanda birçok farklı yöntem geliştirilmiştir (Cai, et al., 2003; Cheema, et al., 2005; Marzolla, et al., 2007; Trunfio, et al., 2007). Bu konuda oldukça anlaşılır ve zengin içerikli bir literatür taraması (Trunfio, et al., 2007) çalışmasında bulunmaktadır. Bu bölümde üç farklı sınıfta incelenen P2P sistemler tabanlı örgü kaynak tarama yöntemlerinden bazıları anlatılmakta, geliştirilen sistemlerin çözümlemeleri yapılarak karşılaştırmaları verilmektedir.

2.2.2.1 Yapılandırılmamış P2P sistemler tabanlı örgü kaynak taraması

Yapılandırılmamış P2P sistemleri, eşler arası sistemlerin ilk örneklerinden olarak gösterilmektedir. Bu tip sistemlerde katılımcılar özel bir ağ ilingesi oluşturmamaktadırlar. Bunun yerine her bir düğüm sisteme komşuları aracılığı ile katılmaktadır. Yapılandırılmamış P2P sistemleri tabanlı örgü kaynak taraması yöntemlerinde (Filali, et al., 2008; Iamnitchi and Foster, 2003), her bir kaynak P2P sisteminin bir katılımcısına denk gelmektedir ve her bir katılımcı kendi kaynak özelliklerini sunmakla görevlidir. Örneğin Iamnitchi ve Foster (Iamnitchi and Foster, 2003) çalışmasında, düğümlerin yapılandırılmamış bir ilinge oluşturduğu P2P tabanlı bir örgü kaynak tarama yöntemi geliştirmişlerdir. Bu algorithmada düğümler istek olması halinde kaynak bilgilerini ağa yayınlamaktadırlar. Filali ve arkadaşları, (Filali, et al., 2008) çalışmasında yapılandırılmamış P2P tabanlı bir örgü kaynak tarama yöntemi geliştirmişlerdir. Kaynak düğümler kaynak bilgilerini dönemsel olarak komşularına göndermektedir. Komşulardan gelen kaynak bilgileri düğümlerin önbelleklerinde saklanmaktadır. Önbellekteki bilgiler sorguların aday düğümlere yönlendirilmesine yardımcı olmak amacıyla kullanılmaktadır.

Yapılandırılmamış P2P tabanlı örgü kaynak tarama yöntemlerinin temel özellikleri göz önüne alındığında çoğu çalışmada sorgular taşıma yöntemi kullanılarak yayılmaktadır. Bu sebeple bu alandaki çalışmaların çoğunun mesaj karmaşıklığı E ağdaki bağlantı sayısı olacak şekilde $O(E)$ 'dir. Buna karşı, bu yaklaşımda kaynaklar ve kaynak dizinleri ağa yayılmış olması sebebiyle bu yöntemdeki algoritmalar devingenliğe karşı dayanıklıdır. Sorguların ağda paralel olarak yayıldığı sistemlerde hatalı çalışan düğümlerin sorguların yayılmasına engel olmamasına karşın, sorguların her adımda tek bir düğüme gönderildiği sistemlerde tekil hata noktası oluşumuna sebep olmaktadır. İncelenen yapılandırılmamış P2P tabanlı yöntemler zaman aşımının kullanılması sonucu hatalı-kabul problemi içermektedir. Bunun sonucunda aranan özellikteki kaynaklar örgüde bulunmalarına rağmen sorgu sonucu olumsuz olarak dönebilmektedir. Zaman aşımının kullanılmadığı veya ağın büyüklüğüne göre çok yüksek tutulduğu çalışmalarda ise algoritmanın sonuca yakınsaması çok uzun sürebilmektedir. Bu alanda incelenen çalışmaların tümü çoklu öznitelik, aralıklı öznitelik ve devinimli öznitelik sorgularının tümünü desteklemektedir.

2.2.2.2 Süper-uç P2P sistemler tabanlı örgü kaynak tarama

Süper-uç tabanlı P2P sistemleri, yapılandırılmamış P2P sistemlerinin ölçeklenebilirliğinin arttırılabilmesi için geliştirilmiş sistemlerdir. Süper-uç sistemlerinde ortam içerisindeki bazı düğümler (süper uç) ayrıcalıklı hale getirilmekte ve diğer düğümlerden farklı görevler üstlenmektedirler (Yang and Garcia-Molina, 2003). Süper-uç P2P sistemleri tabanlı örgü kaynak tarama yöntemlerinde süper uç düğümleri genellikle örgü sistemindeki bir grup düğümün kaynak bilgisinin yönetilmesinden sorumludurlar. Bu sınıftaki yöntemlere ilişkin literatürde birçok çalışma önerilmiştir (Marzolla, et al., 2005; Mastroianni, et al., 2005; Puppini, et al., 2005). Mastroianni ve arkadaşları (Mastroianni, et al., 2005) çalışmasında bazı düğümlerin kaynak düğümler ile ilgili katalog hizmeti sunduğu süper-uç tabanlı bir örgü kaynak tarama yöntemi geliştirmişlerdir. Bu çalışmada her kaynak düğüm bir süper-uç düğümünün yönetimindedir ve kaynak bilgilerini bu düğümlerle paylaşır ve kaynak taraması sorguları süper-uç düğümleri tarafından cevaplanmaktadır. Puppini ve arkadaşları (Puppini, et al., 2005) çalışmasında süper-uç tabanlı bir örgü bilgi servisi geliştirmişlerdir. Bazı düğümleri süper-uçlar olarak belirledikten sonra süper-uçların komşuluklarına bakarak kümeler oluşturmuşlardır. Her bir süper-uç kendi kümesindeki kaynak bilgilerinden sorumlu tutulmuştur. Marzolla ve arkadaşları (Marzolla, et al., 2005) çalışmasında örgü içerisindeki sanal kümeler için dizin servisi görevi gören Çalışma Yüğü

Yönetim Sistemini (ÇYYIS) geliştirmişlerdir. Tanımlanan ÇYYIS'leri bir araya gelip bir ağaç yapısı oluşturmaktadırlar. Her bir ÇYYIS komşularının kaynak bilgilerini saklamakla görevlidir.

Süper-uç tabanlı örgü kaynak tarama yöntemlerinin çoğunluğu sorguların süper-uçlar arasında yayılması amacıyla taşıma tabanlı yönlendirme kullanmaktadır. Bu yaklaşımda süper-uçların kullanılması taşıma tabanlı yönlendirme yapılacak etki alanının küçültülmesi sayesinde yapılandırılmamış yaklaşımlara göre daha etkin olmaktadır. Fakat süper-uçların kullanımı tekil hata noktaları oluşmasına sebep olmaktadır. Bu sebepten dolayı süper-uç düğümlerinin devingenliği bu alandaki algoritmaları kötü yönde etkilemektedir. Ayrıca yöntemlerin çoğu devingen öznitelik tipindeki sorguları da desteklememektedir. Buna karşılık incelenen çalışmaların tümü çoklu öznitelik ve aralıklı öznitelik sorgularını desteklemektedir.

2.2.2.3 Yapılandırılmış P2P sistemler tabanlı örgü kaynak tarama

Yapılandırılmış P2P sistemleri, Dağıtık Kodlama Tablolarının (DKT) kullanımı ile birlikte, eşler-arası sistemlerin son dönemlerinde, yaygın hale gelmiştir. Yapılandırılmış P2P sistemlerinde dizini oluşturulacak bilgi kodlanarak belirli düğümlere atanmaktadır (El-Ansary, 2003). Daha sonra, bilgi aranacağı zaman, DKT kullanımı yardımı ile sorgu aranan bilginin dizinlendiği düğüme yönlendirilme ve aranan bilgiye erişilmektedir. Yapılandırılmış P2P sistemleri tabanlı kaynak tarama yöntemlerinde (Andrzejak and Xu, 2002; Cai, et al., 2003; Oppenheimer, et al., 2004; Spence and Harris, 2003; Talia, et al., 2007) DKT'ler kaynak bilgilerinin dizinlenmesinde kullanılmaktadır. Bunun sonucu olarak, kaynak taraması DKT'ler üzerinden gerçekleşmektedir. Örneğin, Cai ve arkadaşlarının sunduğu (Cai, et al., 2003) çalışmasında, MAAN adı altında, yapılandırılmış P2P tabanlı bir örgü kaynak tarama yöntemi geliştirilmiştir. MAAN, Chord P2P sistemi üzerine geliştirilmiş bir sistemdir. Her bir MAAN düğümü Chord sisteminin bir parçası olarak çalışmaktadır. Kaynak bilgileri Chord Anahtar Uzayına eşlenmiştir. Her düğüm kendi anahtar uzayına denk gelen kaynakların bilgilerinin yönetiminden sorumludur. Andrzejak ve arkadaşları (Andrzejak and Xu, 2002) çalışmasında CAN P2P sistemi tabanlı bir örgü kaynak tarama yöntemi geliştirmişlerdir. Yazarlar bu çalışmada CAN sistemini aralıklı öznitelik sorgularını destekleyebilecek şekilde genişletmişlerdir. Örgüdeki kaynakların her biri kendi özniteliklerine uyan bir grup öznitelikle eşleşmiştir. Örgüdeki bir grup düğüm CAN sistemini oluşturmaktadır. Oppenheimer ve

arkadaşları (Oppenheimer, et al., 2004) çalışmasında SWORD adı altında yapılanmış P2P tabanlı bir örgü kaynak tarama yöntemi geliştirmişlerdir. SWORD sisteminde *raporlayıcı düğümler* ve *sunucu düğümler* olmak üzere iki tipte düğüm tanımlanmıştır. Raporlayıcı düğümler dönemsel olarak kaynak bilgilerini sunucu düğümlere göndermektedirler. Sunucu düğümler ise kaynak bilgilerini kaydetmekte ve sisteme bir kaynak tarama yöntemi sağlamaktadırlar. Sunucu düğümler birbirlerine Bamboo P2P sistemi kullanılarak bağlanmaktadır ve kaynak bilgilerini DHT kullanarak sorgulamaktadır. Spence ve Harris (Spence and Harris, 2003) çalışmasında XenoSearch adında bir yapılanmış P2P tabanlı örgü kaynak tarama ve atama sistemi geliştirmişlerdir. Geliştirilen algoritma Pastry P2P sistemi (Rowstron and Druschel, 2001) üzerinde geliştirilmiştir. Bu çalışmada sunucu düğümler ağaç yapısı oluşturacak şekilde düzenlenmiştir. Kaynak düğümler ağacın yapraklarında bulunacak şekilde düzenlenmişlerdir. Ağaç içerisindeki düğümler ise bilgi akışını sağlamakta ve dönemsel olarak kaynak düğümlerin bilgilerini toplamaktadırlar. Talia, Trunfio ve Zeng (Talia, et al., 2007) çalışmasında geniş ölçekli örgü sistemleri için DKT tabanlı bir kaynak tarama yöntemi geliştirmişlerdir. Bu çalışmada önerilen sistem Chord tabanlı bir algoritmadır. Talia ve arkadaşları Chord sistemini çoklu DKT kullanacak şekilde geliştirerek çoklu öznitelik sorgularını destekleyebilecek özellikler kazandırmışlardır.

Bu bölümde incelenen yapılandırılmış P2P tabanlı örgü kaynak tarama algoritmaları ilingsel yapılar kullanmaları sebebiyle diğer yaklaşımlara göre daha düşük karmaşıklıklara sahiptir. Geliştirilen yöntemlerin çoğunda tüm kaynak düğümler kaynak taramasına katılmaktadır, bu da kuramsal olarak tüm düğümlerin eşit yük dağılımına sahip olmaları anlamına gelmektedir. Bu sebeple incelenen yöntemlerin çoğunda darboğaz problemi bulunmamaktadır. Buna ek olarak, sorguların yapısal bir ilinge üzerinde arama uzayının her yerine yayılmaları zaman aşımı kullanımına olan ihtiyacı ortadan kaldırmakta, bu da hatalı-kabul probleminin oluşmasına engel olmaktadır. Her ne kadar yapılandırılmış P2P sistemlerinin temel özellikleri sebebiyle çoklu öznitelik, devinimli öznitelik ve aralıklı öznitelik sorguları desteklenmemekte ise de, geliştirilen çalışmaların hemen tümü farklı tiplerdeki sorguların desteklenebilmesi için kabul edilir çözümler üretmişlerdir.

2.2.2.4 Karşılaştırma

Önceki kısımlarda farklı sınıflarda incelenen P2P sistemleri tabanlı kaynak

tarama yöntemleri incelenmişti. Bu kısımda farklı sınıflardaki P2P tabanlı kaynak tarama yöntemleri arasında, belirlenmiş değerlendirme ölçütlerine göre bir karşılaştırma sunulmaktadır. Karşılaştırma özeti çizelge 2,2’de gösterilmektedir.

Çizelge 2.2 Eşler-arası sistemler tabanlı kaynak tarama yöntemleri karşılaştırma özeti

	Yapılandırılmamış P2P	Süper-uç P2P	Yapılandırılmış P2P
Ölçeklenebilirlik	Yüksek karmaşıklık sebebiyle ölçeklenebilir değil	Darboğaz problem sebebiyle ölçeklenebilir değil	Ölçeklenebilirliği yüksek
Devingenlik	Her bir düğümün sistemde rol oynaması sebebiyle devingenliğe dayanıklı	Süper-uçların devingenliği halinde başarımı düşük	Devingenlik halinde DKT’nin güncellenme gereksinimi sebebiyle başarımı düşük
Güvenilirlik	KT sorgularının etki alanının kısıtlanması sebebiyle güvenilirliği düşük	Tekil hata noktaları sebebiyle güvenilirliği düşük	Güvenilirliği yüksek
Aralıklı Özellikleri Tarayabilme	Destekleniyor	Destekleniyor	Karmaşık yöntemler kullanılması halinde destekleniyor
Çoklu Özellikleri Tarayabilme	Destekleniyor	Destekleniyor	Karmaşık yöntemler kullanılması halinde destekleniyor
Devingen Özellikleri Tarayabilme	Her zaman güncel sonuçlara erişilebilmesi sebebiyle destekleniyor	Süper-uçlarda bulunan verinin belirli aralıklarda güncellenmesi sebebiyle desteklenmiyor	Karmaşık yöntemler kullanılması halinde destekleniyor

Örgü kaynak taraması için incelenen her üç farklı P2P yaklaşımının (yapılanmamış P2P, süper-uç, yapılanmış P2P) kendilerine ait yarar kazanımları

ve yarar yitimleri bulunmaktadır. Yapılan incelemelere bakıldığında yapılanmamış P2P sistemleri tabanlı çalışmalar farklı tiplerde sorguların ihtiyaç duyulabileceği küçük ölçekli ve devinimli örgü sistemleri için uygun oldukları söylenebilmektedir. Ölçeğin daha geniş olduğu ve hatalı-kabul probleminin onulmaz yanlış olduğu durumlarda ise diğer yaklaşımlar araştırılmalıdır. Süper-uç sistemleri tabanlı yöntemlerin ise daha çok süper-uç düğümlerinin güvenilirliğinin temin edildiği orta ölçekli örgü sistemleri için uygun oldukları, devinimli öznitelik sorgularının kullanıldığı ve hatalı-kabul probleminin ciddi sorunlara sebep olduğu sistemler için ise uygun olmadıkları düşünülmektedir. Öte yandan yapılandırılmış P2P tabanlı sistemlerin ise devingenliğin az olduğu ve sorgu sonuçlarının güvenilirliğinin önemli olduğu geniş ölçekli sistemler için uygun oldukları düşünülmektedir.

2.2.2.5 Sonuçlar

Bu kısımda eşler-arası sistemler tabanlı örgü kaynak taraması yöntemleri ile ilgili bir literatür taraması sunulmuştur. Mevcut çalışmalar kendi içlerinde sınıflandırılmış, her bir sınıf için detaylı incelemeler ve değerlendirmeler sunulmuştur. Sunulan değerlendirmeler ve sınıflar arası karşılaştırmalar dikkate alındığında eşler-arası sistemlerin kaynak taraması için geniş yelpazede çözümler sunduğu anlaşılmaktadır. Her ne kadar her bir sınıftaki çalışmaların kendilerine has güçlü ve zayıf yönleri olsa da, yapılandırılmamış eşler-arası sistemler tabanlı kaynak taraması yöntemlerinin güvenilirlik, devingenliğe dayanıklılık ve farklı türdeki kaynak tarama tiplerini destekleme özellikleri sebebiyle kaynak tarama yönteminin amacına uygun olduğu düşünülmektedir. Ancak bu sistemler ölçeklenebilirlik problemi taşımaktadır. Bu sebeple örgü sistemleri üzerinde kullanılmaları bu haliyle başarımlı kaybına sebep olmaktadır. Öte yandan süper-uç tabanlı kaynak tarama yöntemleri ölçeklenebilirlik anlamında daha güçlü olmalarına rağmen devingenlik ve güvenilirlik konularında daha zayıf kalmaktadırlar. Bu konulardaki en güncel yapılan çalışmalar olan yapılandırılmış eşler-arası sistemler tabanlı kaynak tarama yöntemleri ise karmaşık çözümlerin önerilmesi ile değerlendirme ölçütlerinin hemen tümünü karşılamaktadır. Ancak bu sistemler de sorgu işleminin bir parçası olacak kaynak tarama yöntemi için maliyeti yüksek ve karmaşık sistemlerin ortaya çıkmasına sebep olmaktadır. Bütün bu değerlendirmeler göz önüne alındığında her bir farklı sınıftaki yöntemin kazanımlarını birleştiren bir kaynak tarama yönteminin bu alandaki önemli bir eksiği oluşturduğu düşünülmektedir.

2.2.3 Etmen sistemleri tabanlı örgü kaynak tarama

Son yıllarda özerk ve gezgin etmenler, dağıtık sistemlere özerklik, esneklik ve ölçeklenebilirlik gibi önemli faydalar katmaları sebebiyle çok yaygınlaşmışlardır (Fuggetta, et al., 1998). Etmenlerin bu faydaları örgü kaynak taramasında geniş bir biçimde araştırılmalarına sebep olmaktadır. Etmen tabanlı çalışmalarda (Cao, et al., 2002; Ding, et al., 2005; Kakarontzas and Savvas, 2006; Yan, et al., 2007; Yu, et al., 2006) ana fikir, etmenler arasındaki ortaklık, iletişim kurma ve özerklik gibi özelliklerinden faydalanmaktır. Etmenlerin düzenlenme tipleri ve içsel yapıları verimli kaynak taramasında anahtar rolü oynamaktadır. Etmenlerin dağıtık sistemler için en önemli faydası özerk hareket edebilme kabiliyetleridir. Özerk hareket edebilme, etmenlerin kendi karar düzeneklerine sahip olmaları anlamına gelmektedir. Örneğin bir gezgin etmen ağ üzerinde gezinirken hangi düğüme yönleneceğine kendisi karar verebilir. Buna ek olarak ağdaki düğümlerin farklılıklarından etkilenmeden herhangi bir düğümden çalışabilir, bu şekilde kod gezginliğini sağlar ve örgüdeki düğümlerde kalıcı kod yüklü olması ihtiyacını ortadan kaldırır. Örgü sistemleri, her farklı amaç için kalıcı kod yüklenmesinin yönetsel olarak masraflı olduğu sistemlerdir. Bu sebeple örgü ile etmen sistemlerin tümleştirilmesi ilgi çekici bir yaklaşım olarak nitelenmektedir. Bu bölümde etmen tabanlı örgü kaynak tarama çalışmaları ile ilgili tanımlar, yöntemlerin anlatımları, çözümlemeleri ve verilen ölçütlere göre karşılaştırılmaları verilmektedir. Yöntemler, ağ ilingelerine bakılarak sınıflandırılmışlardır. Bu amaçla, bu alanda yapılan çalışmaları yapılanmamış ağ ilingelerine dayanan yöntemler ve yapılanmış ağ ilingelerine dayanan yöntemler olarak iki farklı sınıf içerisinde incelenmiştir.

2.2.3.1 Yapılanmamış ağ ilingelerine dayanan etmen sistemleri tabanlı kaynak tarama

Bu sınıfta incelenen kaynak tarama yöntemlerinde (Ding, et al., 2005; Jun, et al., 2000; Tang and Huang, 2006; Yu, et al., 2006), etmenler ve etmenlere gönderilen istemler ortama herhangi bir kural olmadan dağıtılmaktadırlar. Örneğin, Ding ve arkadaşları (Ding, et al., 2005) çalışmasında etmen tabanlı bir sezgisel örgü kaynak tarama yöntemi geliştirmişlerdir. Bu çalışmada etmenler örgü üzerindeki düğümlerle eşleştirilmişlerdir ve kullanılabilir düğümlerin bulunabilmesi amacıyla işbirliği yapmaktadırlar. Jun ve arkadaşları (Jun, et al., 2000) çalışmasında etmen sistem tabanlı bir kaynak tarama modeli geliştirmişlerdir. Farklı düğümlerde çalışan etmenler birbirlerinden dağıtık

farkındalık (*distributed awareness*) adı verilen bir yöntem kullanarak haberdar olmaktadırlar. Yu ve arkadaşları (Yu, et al., 2006) çalışmasında gezgin etmen tabanlı bir örgü kaynak tarama yöntemi geliştirmişlerdir. Bu çalışmada, merkezi bir yönetici düğüm, ihtiyaç duyulan kaynak gereksinimlerini araştırarak gezgin etmenleri oluşturmakla sorumludur. Bu düğüm aynı zamanda etmen için yönlendirme kurallarını da tanımlamaktadır. Tang ve Huang (Tang and Huang, 2006) çalışmasında yeni bir örgü kaynak yönetimi algoritması geliştirmişlerdir. Geliştirdikleri sistemde mesajlaşma sunucusu ve kaynak düğüm olmak üzere iki tip düğüm tanımlanmıştır. Mesajlaşma sunucusu gezgin etmenlerin oluşturulmasından sorumludur. Bu tipteki düğümler ihtiyaç duyulan özellikteki kaynakları tarayacak şekilde gezgin etmenler oluşturup yönlendirme kurallarını tanımlamaktadırlar.

Yapılanmamış ağ ilingelerine dayalı etmen tabanlı örgü kaynak tarama yöntemlerinin genelinde darboğaz problemi bulunmamaktadır. Sistemin ölçeklenebilirliğini etkileyen en önemli etmen isteklerin veya etmenlerin dağıtılmasında kullanılan yöntemdir. Durağan etmenlerin kullanıldığı çalışmalarda sorguların dağıtımı mesaj karmaşıklığının yüksek olması sebebiyle ölçeklenebilirliği kısıtlı olan taşıma tabanlı yönlendirmeler ile gerçekleştirilmektedir. Buna karşı gezgin etmenlerin kullanıldığı çalışmalarda, özerklik ve kendi kendine karar verebilme yetenekleri sebebiyle ölçeklenebilirliği arttıran daha verimli yönlendirme yöntemleri kullanılmıştır. Örgü sistemlerinin devingenlikleri göz önüne alındığında ise, durağan etmenlerde kullanılan taşıma tabanlı yönlendirmelerin sorguları paralel olarak birden çok nokta üzerinden dağıtması sebebiyle tekil hata noktası problemi ortadan kalkarken, gezgin etmen tabanlı yaklaşımlarının bazılarında kullanılan tekil yönlendirme patikadaki düğümler tekil hata noktası oluşturmaktadırlar. Her iki yaklaşımda da geliştirilen yöntemlerin çoğunda *Zaman Aşımı (TTL)* kullanılmaması bu yaklaşımlarda hatalı-kabul problemine engel olmaktadır. Fakat bu yaklaşım geniş ölçekli sistemlerde sorgunun yanıtlanmasının çok uzun sürmesine sebep olmaktadır. Bunlara rağmen incelenen algoritmaların çoğu çoklu öznitelik, aralıklı öznitelik ve devinimli öznitelik sorgularının tümünü desteklemektedir.

2.2.3.2 Yapılanmış ağ ilingelerine dayalı etmen sistemleri tabanlı örgü kaynak tarama

Bu sınıfta incelenen çalışmalarda (Kakarontzas and Savvas, 2006; Manvi, et al., 2005; Puh, et al., 2007; Yan, et al., 2007), etmenler kaynak bilgisinin yönetimi

için sanal yapılar oluşturmaktadırlar. Kaynak taramasında kullanılan sorgular etmenler tarafından incelenip belirli bir grup kaynaktan sorumlu olan etmenlere yönlendirilmektedirler. Örneğin, Yan ve arkadaşları (Yan, et al., 2007) çalışmasında kaynakların sanal yapılar (SY) oluşturduğu bir örgü kaynak tarama sistemi geliştirmişlerdir. Her SY bir genel dizin sunucusu ve kaynak düğümlerden oluşmaktadır. Bu sistemde her düğüm hem sorguların gönderildiği istemci, hem de kaynak bilgilerinin yönetildiği sunucu olarak görev yapmaktadır. Kakarontzas ve Savvas (Kakarontzas and Savvas, 2006) çalışmasında örgü kaynak tarama ve seçme için etmen tabanlı bir sistem geliştirmişlerdir. Bu çalışmada örgü kullanıcıları istemci etmenler tarafından temsil edilmektedir. İstemci etmenler kaynak taramasını kaynak bilgilerinin kaydedildiği temsilcileri sorgulayarak gerçekleştirmektedir. (Manvi, et al., 2005) çalışmasında etmen tabanlı bir kaynak atama sistemi (ARAM) geliştirilmiştir. Manvi ve Brije bu çalışmada örgü sistemini kümelemektedir. Her kümenin bir yerel küme sunucusu bulunmaktadır ve bu sunucudaki etmenler kendi kümesindeki kaynakların bilgilerinden sorumlu tutulmaktadır. Puh ve arkadaşları (Puh, et al., 2007) çalışmasında çoklu-etmen tabanlı bir kaynak tarama yöntemi geliştirmişlerdir. Her bir etmen kendi etki alanındaki kaynakların yönetiminden sorumludur.

Yapılanmış ağ ilingelerine dayalı etmen tabanlı örgü kaynak tarama yöntemlerinin çoğunda yükün birden çok düğüme dağıtılması sebebiyle darboğaz problemi bulunmamaktadır. Buna ek olarak etmenlerin yönlendirilmesi de yapılanmış ağ ilingeleri üzerinde verimli yöntemler kullanılarak yapılmaktadır. Fakat bu özellik beraberinde tekil hata noktaları gibi bazı sorunlar da getirmektedir. Örgü ortamının devingen olduğu durumlarda etmenlerin yönlendirme patikaları üzerinde bulunan düğümlerin devingenliği sorguların ağda kaybolmasına sebep olabilmektedir. Bunun yanı sıra büyük sayıda bir kümeyi yöneten bir düğümün hatalı çalışması örgüde erişilebilir durumda olan birçok düğümün erişilememesine sebep olabilmektedir. Bu sınıfta incelenen bütün algoritmalar çoklu öznitelik ve aralıklı öznitelik sorgularını desteklemektedir. Devininimli öznitelik sorguları da çoğu çalışmada etmenlerin özerklik özelliklerinden faydalanılarak desteklenmektedir.

2.2.3.3 Karşılaştırma

Etmen sistemleri kullanan örgü kaynak tarama yöntemlerine ilişkin karşılaştırma özeti çizelge 2.3'te gösterilmektedir. Yapılanmamış ağ ilingelerine dayalı etmen tabanlı kaynak tarama çalışmalarının yüksek karmaşıklık değerleri

sebebiyle diğer sınıfta incelenen çalışmalara göre daha az ölçeklenebilir olduğu düşünülmektedir. Ancak devingenliğe karşı daha fazla dayanıklılık göstermektedirler. Her iki sınıfta incelenen yöntemlerin de güvenilirliklerinin düşük olduğu gözlemlenmiştir. Bunun sebebi, yapılandırılmamış ağ ilingelerine dayalı yöntemlerdeki taşıma mesajı etki alanının kısıtlanması, yapılanmış ağ ilingelerine dayalı yöntemlerde ise tekil hata noktaları olarak listelenmektedir. Her iki sınıfta incelenen yöntemler farklı türdeki KT ihtiyaçlarını karşılamaktadır.

Çizelge 2.3 Etmen sistemler tabanlı kaynak tarama yöntemleri sınıflar arası karşılaştırma özeti

	Yapılanmamış Ağ İlingesine Dayanan Etmen Tabanlı Yöntemler	Yapılanmış Ağ İlingesine Dayanan Etmen Tabanlı Yöntemler
Ölçeklenebilirlik	Yüksek zaman ve mesaj karmaşıklığı sebebiyle ölçeklenebilir değil	Yükün sıradüzensel dağıtılması sayesinde ölçeklenebilirliği yüksek
Devingenlik	KT sorgularının koşut dağıtılması sebebiyle çoğu çalışmada devingenliğe dayanıklı	Yapıdaki bozulmanın çok miktarda düğümü etkilemesi sebebiyle devingenliğe dayanıklı değil
Güvenilirlik	KT sorgularının etki alanının sınırlandırılması sebebiyle güvenilirliği düşük	Tekil hata noktası problemi sebebiyle güvenilirliği düşük
Aralıklı Özellikleri Tarayabilme	Destekleniyor	Destekleniyor
Çoklu Özellikleri Tarayabilme	Destekleniyor	Destekleniyor
Devingen Özellikleri Tarayabilme	Destekleniyor	Destekleniyor

2.2.3.4 Sonuçlar

Etmen tabanlı örgü kaynak tarama yöntemleri içerisinde incelenen her iki sınıftaki yöntemlerin de birbirlerine göre yarar kazanımları ve yarar yitimleri

bulunmaktadır. Yapılan çözümlerlere dayanarak yapılanmamış sistemlerde ölçeklenebilirlik ve devingenliğe dayanıklılık özelliklerinin hangi tip etmen kullanıldığına bağlı olarak değiştiği söylenebilir. Buna göre geniş ölçekli ve devingenliği nispeten daha az olan ortamlarda özerk hareket edebilme ve yönlendirme özelliklerine sahip olmaları sebebiyle gezgin etmen tabanlı sistemler daha tercih edilebilir durumdadırlar. Ölçeğin daha küçük olduğu ve devingenliğin daha yüksek olduğu sistemlerde ise tekil hata noktası probleminin olmaması sebebiyle durağan etmen tabanlı sistemler tercih edilebilir. Buna karşı yapılanmış ağ ilingesine sahip olan yöntemlerde hangi tip etmenin kullanıldığından bağımsız olarak sıradüzensel yapı kullanımı ve azalan karmaşıklıklar sebebiyle sistem geniş ölçekli ortamlara yapılanmamış olanlara göre daha uygundur. Fakat bu sistemler de genellikle tekil hata noktası probleminden etkilenmektedirler.

2.2.4 Kaynak tarama yöntemlerinin genel değerlendirilmesi

Bu kısımda yakın zamanlarda geliştirilen örgü kaynak tarama yöntemleri derinlemesine incelenmiş, farklı sınıflardaki yaklaşımların birbirlerine göre yarar kazanımları ve yarar yitimleri detaylı olarak tartışılmıştır. Farklı yöntemlerin belirlenen değerlendirme ölçütlerine göre karşılaştırma özeti çizelge 2.4'te gösterilmektedir.

Çizelge 2.4 Farklı kaynak tarama yöntemi sınıfları arasında karşılaştırma özeti

	Merkezi / Sıradüzensel Sistemler Tabanlı KT	P2P Sistemler Tabanlı KT	Etmen Sistemler Tabanlı KT
Ölçeklenebilirlik	Küçük ölçekli sistemler için uygun	Geniş ölçekli sistemler için uygun	Küçük ve orta ölçekli sistemler için uygun
Devingenlik	Kaynakların devingenliğine dayanıklı, dizinleme düğümlerinin devingenliğine dayanıksız	Devingen sistemlerde başarımlı düşük	Etmenlerin özerklik özellikleri sebebiyle dayanıklı
Güvenilirlik	Güvenilirlikleri düşük	Güvenilirlikleri yüksek	Güvenilirlikleri düşük
Arahlık Özellikleri Tarayabilme	Destekleniyor	Karmaşık yöntemler kullanılarak destekleniyor	Destekleniyor

Çoklu Özellikleri Tarayabilme	Destekleniyor	Karmaşık yöntemler kullanılarak destekleniyor	Destekleniyor
Devingen Özellikleri Tarayabilme	Çoğunda destekleniyor	Karmaşık yöntemler kullanılarak destekleniyor	Destekleniyor

Yapılan karşılaştırmalarda da görülebileceği üzere her yaklaşımın kendine has özellikleri, faydaları ve eksiklikleri bulunmaktadır. Tezin bu bölümünde, incelenen kaynak tarama yöntemlerinin faydalarını mümkün olduğunca içine alıp eksikliklerinden kaçınan örgü kaynak tarama algoritmaları önermeyi hedeflemekteyiz. Günümüz çalışmalarına bakıldığında zaman etmen tabanlı sistemler etmenlerin özerklik gibi özellikleri sebebiyle ilgi çekici oldukları görülmektedir. Fakat önceden tahmin edilemeyen ve ilingesel olmayan hareket desenleri, sorguların çoğunlukla taşıma ile yayıldığı bu çalışmalarda hatalı-kabul problemlerinin ortaya çıkmasına sebep olmaktadır. Öte yandan P2P tabanlı sistemler incelendiğinde, güncel çalışmaların çoğu DHT kullanan yapılanmış P2P sistemleri üzerine yoğunlaşmıştır. DHT kullanımı sorguların etkin biçimde yanıtlanmasına olanak sağlamaktadır. Fakat DHT'lerin de kaynak taraması alanında bazı yarar yitimleri olduğu görülmektedir. Kaynak taramasında DHT'lerin kullanımı çoğu çalışmada devingen öznitelik sorguları ve düğümlerin devingenliği açısından kısıtlayıcı olmaktadır. Devingen özniteliklerin DHT üzerinde güncellenmesinin maliyetli olması bu özniteliklerin DHT kullanılarak taranmasının uygulanabilirliğini düşürmektedir. Bu soruna çözüm bulabilmek amacıyla bu sınıftaki algoritmaların çoğu DHT'nin üzerinde çalıştığı ilingenin yapısal özelliklerini kullanarak etkin yönlendirme ve yayma algoritmaları geliştirmişlerdir. Bu yaklaşım süper-uç tabanlı sistemleri çağırıştırılmaktadır. Süper-uç tabanlı kaynak tarama yöntemleri ilingesel yapıların özelliklerini kullanmaktadır. Fakat bu çalışmaların çoğu süper-uç düğümlerini bir altkümenin özelliklerini tutmak için kullanmaktadır. Bu da süper-uç düğümlerinin tekil hata noktaları oluşturmalarına sebep olmaktadır. Fakat süper-uç sistemleri örgü sistemlerin kümeli yapıları ile kolayca örtüştürülebilmektedirler. Bu sebeple süper-uç sistemlerinde var olan problemlerin ortadan kaldırılması halinde bu sistemlerin örgü kaynak taramasında etkin yöntemler olarak kullanılabileceklerini düşünmekteyiz.

2.2.5 Öz-kararlı kapsayan ağaç oluşturma algoritmaları

Kapsayan ağaç algoritmaları dağıtık uygulamalarda sıkça kullanılan ilinge yapılarıdır. Bir kapsayan ağaç içinde bulunduğu çizgedeki tüm düğümleri içeren ve içerisinde döngü bulunmayan ağaç yapısı olarak tanımlanır. Örgü sistemleri gibi dağıtık sistemlerin boyutlarındaki büyüme ile birlikte kapsayan ağaç gibi ilinge denetimi mekanizmaları büyük ölçüde önem kazanmıştır. Bu kontrol mekanizmaları, dağıtık uygulamaların altında yatan çizgedeki karmaşık bağlantı yapısı sebebiyle ortaya çıkan karmaşıklığı azaltmak amacıyla kullanılmaktadır. Kapsayan ağaçların kullanımı birçok dağıtık algoritmanın etkin biçimde çalışabilmesine olanak sağlamaktadır. Özellikle çoğa gönderim ve tüme gönderim işlemlerini içeren algoritmalarda kapsayan ağaçlar sıklıkla kullanılmaktadır. Son zamanlarda ölçeğin de genişlemesiyle birlikte tüm sistemin bilgisinin tek bir yerde tutulmasının zor olması sebebiyle dağıtık kapsayan ağaç oluşturma algoritmaları önem kazanmıştır. Bu alanda birçok farklı tipte kapsayan ağaç oluşturma algoritması geliştirilmektedir. Birçok çalışma en küçük kapsayan ağaç bulma üzerine yoğunlaşmıştır (Ahuja and Zhu, 1989; Awerbuch, 1987; Gallagher, et al., 1983; Lien, 1988). En küçük kapsayan ağaç yapısı, bir çizge üzerinde oluşturulabilecek kapsayan ağaçlar içerisinde kenar ağırlıkları toplamı en az olan kapsayan ağaç olarak tanımlanmaktadır. En küçük kapsayan ağaçlar iletişim maliyetinin azaltılması için oldukça kullanışlı yapılardır. Bunun yanı sıra en küçük dereceli kapsayan ağaç oluşturma (Blin, et al., 2009), en küçük çaplı kapsayan ağaç oluşturma (Butelle, et al., 1995) gibi farklı algoritmalar da değişik amaçlara yönelik olarak dikkat çekmektedir.

Klasik dağıtık kapsayan ağaç oluşturma algoritmalarının yanında birçok çalışma devingenlik ve aksaklığa dayanıklılığı da irdlemiştir. Bu kapsamda, öz kararlı kapsayan ağaç oluşturma algoritmaları geliştirilmiştir (Gartner, 2003). Öz kararlılık kavramı ağdaki devingenliklere rağmen kapsayan ağaç yapısının yeni baştan oluşturulmaya gerek kalmadan geçerliliğini koruması esasına dayanmaktadır. Daha genel olarak açıklamak gerekirse, öz kararlılık dağıtık sistemlerin aksaklıklar halinde sonlu zaman içerisinde istenen konuma geri dönebilmesi olarak tanımlanmıştır. Bu kavram ilk olarak 1974 yılında Dijkstra (Dijkstra, 1974) tarafından ortaya atılmıştır. Öz kararlı algoritmalarda her düğüm yerel değişkenler tutmaktadır. Düğümler bu değişkenlere ve komşularının değişkenlerine göre hareket etmektedir. Düğümlerin yerel değişkenleri yerel konumlarını, tüm düğümlerin yerel konumları ise sistemin konumunu tanımlamaktadır. Hedef, sistemin konumunu geçerli konumda tutmaktır. Öz

kararlılık, devingenliğin yüksek olduğu örgü sistemleri gibi ortamlarda birçok avantaj sağlamaktadır, bu sebeple örgü üzerinde kapsayan ağaç oluşturma algoritmalarında kullanılmaları birçok durumda uygundur.

2.2.5.1 Yakın zamanda yapılan çalışmalar

Literatürde birçok öz-kararlı kapsayan ağaç oluşturma algoritması bulunmaktadır (Afek, et al., 1991; Antonoiu and Srimani, 1995; Antonoiu and Srimani, 1997; Baala, et al., 2003; Dolev, 1993; Gupta and Srimani, 2003; Herault, et al., 2006; Kosowski and Kuszner, 2006; Pan, et al., 1999). Bu çalışmalardan günümüzde sıklıkla karşılaşılanlar aşağıdaki şekilde özetlenebilmektedir. Afek ve arkadaşları (Afek, et al., 1991) bilgisayar ağlarında kullanılmak üzere verimli bellek kullanan öz kararlı bir kapsayan ağaç algoritması geliştirmişlerdir. Bu tasarımda her düğüm başlangıç olarak kendisini kök ilan etmektedir. Daha sonra düğümler komşuluklarını tarayıp daha büyük köke sahip bir ağaç olup olmadığını araştırmaktadır. Daha büyük köke sahip bir ağaç bulunması halinde düğüm ebeveynini belirler ve kapsayan ağaca katılmaktadır. Dolev ve arkadaşları (Dolev, 1993) çalışmalarında tekdüze BFS tipinde kapsayan ağaç oluşturan bir algoritma geliştirmişlerdir. Tasarımlarında her düğüm algoritmayı komşuluklarını göz önüne alarak başlatır. Eğer düğümün kimlik numarası kendi komşuluğu içerisinde en büyük kimlik numarası ise düğüm kendini kök düğüm olarak ilan etmektedir. Algoritmanın sonunda bir BFS kapsayan ağaç oluşmaktadır. Butelle ve arkadaşları (Butelle, et al., 1995) çalışmasında ağırlıklı kenarlara sahip bir çizge üzerinde en küçük çaplı kapsayan ağaç bulan öz kararlı bir algoritma tasarlamışlardır. Tasarımları çizge üzerindeki orta düğüm üzerine odaklanmıştır. Orta düğüm ağacın kök düğümü olarak kullanılmaktadır ve öz kararlı olarak belirlenmektedir. Antonoiu ve Srimani (Antonoiu and Srimani, 1995) çalışmasında gelişigüzel bir kapsayan ağaç oluşturan öz kararlı bir algoritma geliştirmiştir. Tasarımlarında kapsayan ağacı oluştururken referans noktası oluşturma amacıyla bir kök düğümün var olduğunu varsaymışlardır. (Antonoiu and Srimani, 1997) çalışmasında ise Antonoiu ve Srimani en küçük kapsayan ağacı bulan bir öz kararlı algoritma tasarlamıştır. Algoritmanın sonunda çizge üzerinde bir en küçük kapsayan ağaç oluşturulmaktadır. Pan ve arkadaşları (Pan, et al., 1999) çalışmasında öz kararlı en büyük bulma yöntemi tabanlı öz kararlı bir kapsayan ağaç oluşturma algoritması geliştirmişlerdir. Tasarımlarında önce en büyük kimlikli düğümü bulup daha sonra her düğümün bu düğüme olan uzaklıklarını hesaplamaktadırlar. Ebeveyn ilişkileri bu uzaklık değerlerine göre belirlenmektedir. Bu algoritmanın sonunda BFS (*ing.*

breadth first search) tipinde bir kapsayan ağaç oluşturulmaktadır. Baala ve arkadaşları (Baala, et al., 2003) çalışmasında gelişigüzel yürüme yöntemi tabanlı bir öz kararlı kapsayan ağaç oluşturma algoritması geliştirmişlerdir. Gelişigüzel yürüyüş bölümü bağımsız gezgin etmenler tarafından gerçekleştirilmektedir. Kök düğüm algoritmanın sonunda belirlenmektedir, dolayısı ile bu tasarım önceden belirlenmiş bir kök düğüm olmasını gerektirmemektedir. Gupta ve Srimani (Gupta and Srimani, 2003) çalışmasında iki adet öz kararlı kapsayan ağaç oluşturma algoritması geliştirmişlerdir. İlk algoritmalarında tasarsız bir ağ üzerinde en kısa patikalı kapsayan ağaç oluşturmaya hedeflemişlerdir. İkinci algoritmalarında ise en küçük kapsayan ağaç bulmayı hedeflemişlerdir. (Herault, et al., 2006) çalışmasında, Herault ve arkadaşları geniş ölçekli sistemler için öz kararlı bir kapsayan ağaç oluşturma algoritması geliştirmişlerdir. Algoritmanın sonunda en büyük kimlik numarasına sahip olan düğüm kök düğümü olacak şekilde bir kapsayan ağaç oluşmaktadır. (Kosowski and Kuszner, 2006) çalışmasında, Kosowski ve Kuszner çalışmasında iki öz kararlı kapsayan ağaç oluşturma algoritması geliştirmiştir. Çalışmalarında kapsayan ağaç oluşturmaya başlamak için bir kök düğümü seçmişlerdir. İlk algoritmalarında her düğüm en küçük kimlik numarasına sahip komşusunu ebeveyni olarak seçmektedir. Kök düğümü her zaman 0 kimlik numarasına sahiptir. İkinci algorithma ise düğümler ebeveyn seçiminin yapılabilmesi için kimlik numarasından farklı bir değişken tutmaktadır. Blin ve arkadaşları (Blin, et al., 2009) çalışmasında en küçük dereceli kapsayan ağaç bulan öz kararlı bir algoritma geliştirmişlerdir. Blin ve arkadaşları (Afek, et al., 1991) çalışmasında geliştirilen algorithmadan esinlenmişlerdir, ek olarak her döngüde kapsayan ağacın derecesini azaltan bir derece azaltma eklentisi geliştirmişlerdir.

2.2.5.2 Sonuçlar

İncelenen çalışmaların büyük bir bölümü ağdaki düğümlerin kimlik bilgilerinden faydalanmaktadır (Afek, et al., 1991; Dolev, 1993; Herault, et al., 2006; Kosowski and Kuszner, 2006; Pan, et al., 1999). Diğer çalışmalar ise kenar ağırlıkları (Antonoiu and Srimani, 1997), özel yetkiye sahip düğümler (Antonoiu and Srimani, 1995; Gupta and Srimani, 2003) veya rastgele yürüyüş (Baala, et al., 2003) gibi farklı yöntemler ile kapsayan ağaç oluşturmaktadır. Her ne kadar burada incelenen çalışmalar öz-kararlı kapsayan ağaç oluşturma problemine farklı çözümler sunsalar da, kapsayan ağaç üzerinde tüme gönderim ve tümünden toplama işlemlerinin eniyilemesi amacıyla hazırlanmış bir çalışmaya bilgimiz dâhilinde rastlanamamıştır. Bu amaç gözetilerek, kapsayan ağaç oluşturmada

kullanılabilecek daha etkin deęiřtirgeler bulunabilmektedir. Örneęin düęümlerin kimlik bilgileri yerine komřuluk dereceleri kullanılarak elde edilecek tekil kimlik numaralarının kullanılmasının oluřacak kapsayan aęaę üzerinde tüme gönderim ve tümenden toplama iřlemlerinin başarımını arttıracadı düşünölmektedir.

2.2.6 Kaynak taraması algoritmaları için sonuçlar

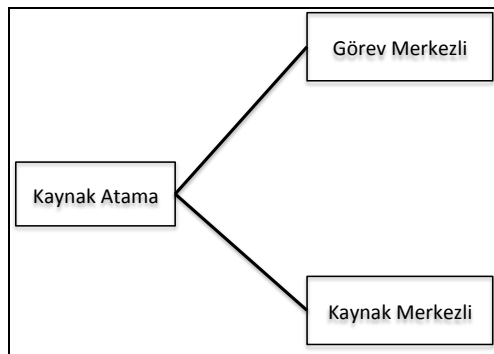
Önceki kısımlarda incelenen kaynak tarama yöntemlerinin her birinin kendine özgü fayda kazanımları ve fayda yitimleri bulunmaktadır. Çalışmaların birbirleriyle ortak fayda kazanımlarının ve yitimlerinin ortaya çıkarılabilmesi amacıyla mevcut çalışmalar sınıflandırılarak incelenmiştir. Farklı sınıflardaki kaynak tarama yöntemlerinin fayda kazanımlarının birleştirilmesiyle elde edilebilecek olan kaynak tarama yönteminin araştırılmasının bu alanda bir yenilik getireceęi düşünölmektedir. Örneęin eşler arası sistemler tabanlı KT yöntemlerinin verimli bir öz-kararlı ilinge denetimi bile eşleştirilerek elde edilecek olan KT yönteminin tanıtılan deęerlendirme ölçütlerinin tamamını karşılayabileceęi düşünölmektedir. Bu konuda yapılan literatür taraması sonuçları dikkate alındığında, bilğimiz dahilinde örgü sistemlerinin özelliklerinden ve sorgu iřlemesinin gereksinimlerinden kaynaklanan sorunlara odaklı bir kaynak tarama çalışmasına rastlanılamamıştır. Bu sebeple, bu tezin ilk bölümünde bir çapraz katman yapısı önerilerek ilinge denetimi ve onun üzerinde kaynak tarama algoritması olacak şekilde bir kaynak tarama yöntemi önerilmektedir.

Bu amaçla, ölçeklenebilirlik problemine çözüm olarak verimli bir ilinge denetimi önerilmektedir. Devingenlik problemine çözüm olarak önerilecek ilinge denetiminin öz-kararlı olması kararlařtırılmıştır. Güvenilirlik ile ilgili olarak örgü sistemindeki tüm düęümlerin kaynak taramasına katılması sağlanacaktır. Çoklu öznitelik, aralıklı öznitelik ve devingen öznitelik sorgularının desteklenebilmesi amacıyla ise kaynak tarama sorgularının her bir kaynaęın kendisi tarafından cevaplanması sağlanacaktır.

2.3 Örgü Sistemlerde Sorgu İřleme İçin Kaynak Ataması

Kaynak ataması, görev kümesinden kaynak kümesine birebir olmayan (*ing. non-injective*), örten olmayan (*ing. non-surjective*) ve çok deęerli (*ing. multi-valued*) olan bir fonksiyon olarak tanımlanabilmektedir. Bir önceki adım olan kaynak taraması aşamasında görevleri yerine getirebilecek yeteneęe sahip aday kaynaklar belirlendikten sonra, mevcut görevlerin verimli bir biçimde

yürütülebilmesi için kaynak ataması yapılması gerekmektedir. Kaynak atamasında: (i) kaç adet kaynağın kullanılacağı, (ii) hangi görevlerin hangi kaynaklarda yürütüleceği ve (iii) görevlerin yürütülmesi sırasında kaynakların devingenliğinden kaynaklanan değişimlerde nasıl hareket edileceği belirlenmektedir. Bu konuda, farklı yöntemler kullanılarak bir çok çalışma yapılmıştır (Gounaris, et al., 2004; Slimani, et al., 2004; Gounaris, et al., 2005; Kant and Grosu, 2005; Mandal, et al., 2005; Soe, et al., 2005; Gounaris, et al., 2006; Silva, et al., 2006; Venugopal, et al., 2006; Bose, et al., 2007; Zhao, et al., 2007; Kotowski, et al., 2008; Liu and Karimi, 2008; Gounaris, et al., 2009; Quian'e-Ruiz, et al., 2009; Paton, et al., 2009). Bu alandaki mevcut çalışmalar literatürde genellikle iki sınıf altında incelenmiştir: (i) durağan kaynak ataması ve (ii) devingen kaynak ataması. Durağan kaynak atama yönteminde kaynak ataması başlangıçta gerçekleştirilmekte ve görevlerin yürütülmesi tamamlanana kadar aynı kalmaktadır. Öte yandan, devingen kaynak atamasında kaynak ataması tamamlandıktan sonra görevlerin yürütülmesi gözlemlenip gerekli hallerde kaynak ataması değiştirilmektedir. Ancak örgü sistemlerinin özellikleri göz önüne alındığında devingenlik özelliğinin bir tasarım önkoşulu olması gerektiği düşünülmektedir. Tezin bu kısmında, örgü sistemleri üzerinde sorgu eniyilemesi için kaynak ataması konusu üzerinde bir literatür taraması ve sınıflandırma gerçekleştirilmiştir. Bu anlamda literatürdeki mevcut çalışmalar hedeflerinin kaynak kullanımı eniyilemesi veya görev yürütülmesi eniyilemesi olmasına göre *görev* merkezli ve *kaynak* merkezli kaynak ataması algoritmaları olmak üzere iki farklı sınıfta incelenmiştir. Literatür taramasında kullanılacak olan sınıflandırma şekil 2.2’de gösterilmektedir.



Şekil 2.2 Örgü sistemlerinde sorgu işleme için kaynak atama yöntemleri sınıflandırma özeti

En iyi kaynak atama şeklinin bulunması NP-Complete bir problem olarak tanımlanmaktadır (Fernandez-Baca, 1989; Chihping, 1990), bu sebeple literatürdeki mevcut çalışmalar kaynak atama problemine sorgu çalışma süresini

kısaltacak yaklaşımsal çözümler önermişlerdir. Bu kısımda mevcut kaynak atama yöntemleri için bir literatür taraması sunulmaktadır. Mevcut çalışmalar incelenmekte ve hem niteliksel hem niceliksel ölçütlere göre kıyaslanmaktadır. Niceliksel değerlendirmeler ve karşılaştırmalar için GridSim (Sulistio, et al., 2008) benzetim ortamı kullanılmaktadır. Niteliksel değerlendirme ve karşılaştırmalar için ise önemli olduğunu düşündüğümüz aşağıdaki ölçütler kullanılmaktadır:

- Göreve özgü farklı gereksinimlerin gözönüne alınması: Dağıtık sorgulardaki her farklı görevin farklı gereksinimleri olabilmektedir. Örneğin bir görev için yeterli bellek alanı önemliyken, bir diğer görev için işlemci özellikleri daha önemli olabilmektedir. Bu sebeple her görev için farklı gereksinimlerin dikkate alınması gerekmektedir.

- Görevler arası ilişkilerin dikkate alınması: Dağıtık veritabanı sorgularına ait görevlerden bazıları birbirleriyle haberleşmektedir. Bir başka deyişle, bir görev tarafından üretilen veri bir diğer görev tarafından kullanılabilir. Bu gibi durumlarda görevler arasında veri akışı gerçekleşmektedir. Kaynak atama algoritmalarının, kaynaklar arasındaki ağ bağlantılarının özellikleri de gözetilerek bu durumu göz önüne alması beklenmektedir.

- Yük dengesinin göz önüne alınması: Kaynak atamasında kaynaklar arasında yük dengesinin göz önüne alınması koşut görevlerin tamamlanma sürelerindeki eğilimleri ortadan kaldırmak amacıyla önem taşımaktadır.

- Kaynakların özelliklerinin göz önüne alınması: Örgü sistemleri, türdeş olmayan kaynaklar içermeleri sebebiyle, kaynak atama yönteminin kaynakların değişik özelliklerini göz önüne alması gerekmektedir (işlemci hızı, çekirdek sayısı, ağ bağlantı hızı, kullanılabilir bellek alanı, vs.). Kaynakların başarımlarının tahmin edilmesinde, bu özelliklerin incelenmesindeki detay düzeyi önemli rol oynamaktadır.

- Kaynak atama yordamının ölçeklenebilirliği ve güvenilirliği: Örgü sistemleri geniş ölçekli de devingen olarak değerlendirilmektedir, bu sebeple atanmayı bekleyen görevlerin ve aday kaynakların sayısı çok büyük olabilmektedir. Bu anlamda, kaynak atama yönteminin darboğaz problemi

ve tekil hata noktası problemi bulunmamalıdır.

Bu kısmın geri kalanında, kısım 2.3.1’de örgü sistemlerinde sorgu işleme için kaynak atama yöntemleri ile ilgili sunulmuş güncel literatür taraması çalışmaları sunulmuştur. Kısım 2.3.2’te, kaynak atama yöntemleri sınıflandırmalar ile incelenmekte ve literatür özeti sunulmaktadır. Kısım 2.3.3’te, hataya dayanıklı kaynak atama yöntemleri ile ilgili bir literatür taraması sunulmuştur. Kısım 2.3.4’te ise kaynak atama yöntemleri ile ilgili olarak varılan sonuçlar sunulmaktadır.

2.3.1 Mevcut literatür taraması çalışmaları ve motivasyonlar

Literatürde, örgü sistemleri üzerinde kaynak ataması yöntemlerini inceleyen detaylı literatür özeti çalışmaları bulunmaktadır (Krauter, et al., 2002; Jian and Liu, 2007; Jiang, et al., 2007; Costa and Furtado, 2008; Epimakhov, et al., 2011; Cokuslu, et al., 2012a). Jian ve arkadaşları (Jian and Liu, 2007) çalışmasında örgü üzerinde kaynak atama algoritmalarına dair bir literatür özeti çalışması sunmuşlardır. Bu çalışmada mevcut yöntemler üç sınıfta incelenmiştir: hesaplama dayalı ekonomi tabanlı (*ing. computational economy based*), etmen tabanlı (*ing. agent based*) ve servis odaklı (*ing. service oriented*) kaynak atama yöntemleri. Bu çalışma detaylı bir literatür özeti sunmaktadır, fakat incelenen yöntemlerin değerlendirilmesi için gerekli ölçütler belirtilmemiştir. Bu açıdan incelenen algoritmaların standartlaşmış ölçütlerle değerlendirilmesi ve karşılaştırılması verilmemiştir. Jiang ve arkadaşları (Jiang, et al.) çalışmasında sundukları literatür özetinde örgü sistemlerinin özelliklerini ve değerlendirme ölçütlerini açıkça belirtmişlerdir. Fakat bu çalışma örgü sistemleri üzerinde kaynak ataması yöntemlerini incelerken güvenlik ve aksaklığa dayanıklılık konuları üzerinde yoğunlaşmıştır. Bunun yanı sıra incelenen algoritmalar arasında sınıflandırma ve karşılaştırma verilmemiştir. Krauter ve arkadaşları (Krauter, et al.) çalışmasında örgü sistemleri için kaynak yönetim sistemleri üzerine sınıflandırma ve literatür özeti sunmuşlardır. Sınıflandırmalarını kaynakların düzeni (*ing. machine organization*), kaynak modeli (*ing. resource model*), dağıtım kuralları (*ing. dissemination protocols*), isim uzayı düzeni (*ing. namespace organization*), veri deposu düzeni (*ing. data store organization*), kaynak taraması (*ing. resource discovery*), hizmet kalitesi desteği (*ing. QoS support*), kaynak atama düzeni (*ing. scheduler organization*), kaynak atama politikası (*ing. scheduling policy*), durum tahminlemesi (*ing. state estimation*) ve tekrar kaynak ataması yaklaşımı (*ing. rescheduling approach*) konuları üzerinde

düzenlemişlerdir. Bu konular içerisinde, kaynak atama aşamasını merkezi, sıradüzensel ve dağıtılmış kaynak atama algoritmaları olacak şekilde üç sınıfta incelemişlerdir. (Krauter, et al., 2002) çalışması oldukça detaylı bir çalışma olmasına rağmen değerlendirme ölçütleri ve sınıflar arası karşılaştırmalar verilmemiştir. Costa ve arkadaşları (Costa and Furtado) çalışmasında farklı kaynak atama algoritması sınıfları arasında deneysel değerlendirme ve karşılaştırma sunmuşlardır. Sınıflar incelenirken örgü sistemlerinin özellikleri de göz önüne alınmıştır. Çalışmada kaynak atama algoritmaları merkezi ve sıradüzensel olmak üzere iki sınıfta incelenmiştir. Costa ve arkadaşları sınıflar arası değerlendirme ve kıyaslamalar sunmalarına rağmen, incelenen sınıfların niteliksel ölçütlere göre analizleri ve değerlendirmeleri verilmemiştir. Bir diğer detaylı literatür özeti sunan (Epimakhov, et al., 2011) çalışmasında, Epimakhov ve arkadaşları örgü sistemlerde sorgu eniyileme için kaynak atama yöntemlerini incelemişlerdir. Mevcut çalışmaları klasikten genişletilmiş ve güdülenerek önerilmiş sistemler olacak şekilde iki sınıf altında incelemişlerdir. Her bir sınıf için detaylı incelemelerde bulunulmuş ve benzetim sonuçları ile kıyaslamalar sunulmuştur. Bu kısmın bir bölümünü de kapsayan (Cokuslu, et al., 2012a) çalışmasında Cokuslu ve arkadaşları örgü sistemleri üzerinde sorgu işleme için kaynak atama algoritmaları üzerine bir literatür taraması sunmuşlardır. Çalışmalarında mevcut yöntemleri kaynak merkezli ve görev merkezli çalışmalar olmak üzere iki sınıf altında incelemişler, detaylı tahliller ve değerlendirmeler ile farklı sınıftaki çalışmaları birbirleriyle karşılaştırmışlardır.

2.3.2 Yakın zamanda yapılan kaynak atama çalışmaları

Bu kısımda örgü sistemleri üzerinde sorgu işlemesi için tasarlanmış güncel kaynak atama algoritmaları üzerinde bir literatür özeti sunulmaktadır. İncelenen çalışmalar eniyileme hedeflerine göre sınıflandırılmıştır. Her bir sınıfta incelenen algoritmalar için ilk önce detaylı özet ve değerlendirmeler verilmiştir. Bunu takiben her sınıf için genel bir değerlendirme ve kısım 2.3'te bahsi geçen niteliksel ölçütlere göre karşılaştırma verilmiştir.

2.3.2.1 Görev merkezli kaynak atama algoritmaları

Görev merkezli kaynak atama algoritmalarında soruna kullanıcı tarafından bakılmaktadır. Öncelikli amaç, kaynakların kullanılma durumlarından ziyade, görevleri en kısa sürede tamamlamaktır. Bu amaçla kaynakların işlemci hızı, kullanılabilir bellek alanı, ağ bağlantı hızı gibi temel özellikleri dikkate alınarak

görevler için en güçlü kaynaklar seçilmektedir. Genel olarak temel yaklaşım, kaynaklar için bir değerlendirme fonksiyonu hazırlanıp bu fonksiyonun sonucuna göre görevleri en yüksek değere sahip kaynaklara atamak şeklinde özetlenebilir. Bu sınıfa dâhil literatürde birçok çalışma bulunmaktadır (Bose, et al., 2007; Gounaris, et al., 2004; Gounaris, et al., 2006; Kant and Grosu, 2005; Liu and Karimi, 2008; Mandal, et al., 2005; Silva, et al., 2006; Venugopal, et al., 2006). Örneğin, (Kant and Grosu, 2005) çalışmasında Kant ve Grosu ihale tabanlı (*ing. auction based*) kaynak atama algoritmaları önermişlerdir. Geliştirdikleri modelde her bir kullanıcının çalıştırmaya hazır toplam iş miktarı, görev sayısı ve bütçe olmak üzere değiştirgeleri bulunmaktadır. Benzer şekilde kaynak düğümlerin de işlemci hızı, rezervasyon ücreti, maliyet ve işe göre kazanç gibi değiştirgeleri bulunmaktadır. İhalelerin sonunda kaynak düğümlerle görevler eşleştirilmekte ve görevler çalıştırılmaktadır. Mandal ve arkadaşları (Mandal, et al., 2005) çalışmasında örgü üzerinde iş akışı için bir kaynak atama algoritması önermişlerdir. Çalışmada, aday kaynak düğümler sorgu bileşenlerindeki görevler için beklenen başarımlarına göre puanlanmakta ve kaynak ataması bu puanlamaya göre gerçekleştirilmektedir. Gounaris ve arkadaşları (Gounaris, et al., 2004; Gounaris, et al., 2006) çalışmasında örgü sistemleri üzerinde koşut sorgu işleme için bir kaynak atama yöntemi geliştirmişlerdir. Çalışmada özellikle bölümlenmiş koşut problemi irdelenmiştir. (Bose, et al., 2007) çalışmasında, Bose ve arkadaşları verilen bir koşut sorgu kümesi için verimlik hedefli bir kaynak atama algoritması sunmuşlardır. Çalışmada sorgu planlarının gür ağaç (*ing. Bushy Tree*) gösterimlerinden yararlanmışlardır. (Liu and Karimi, 2008) çalışmasında Liu ve Karimi örgü sistemlerde sorgu işleme için bir sorgu eniyileyici geliştirmişlerdir. Önerilen sistemde kaynak taraması, kaynak ataması ve sorgu eniyilemesi adımlarının her biri ayrı irdelenmiştir. Çalışmada, aday kaynaklar belirlendikten sonra bir puanlama fonksiyonu vasıtasıyla kaynak ataması gerçekleştirilmiştir. Silva ve arkadaşları (Silva, et al., 2006) çalışmasında, içerisinde bir kaynak atama biriminin de bulunduğu uyarlanabilir bir koşut sorgu işleme sistemi geliştirmişlerdir. Önerilen algoritmada koşut sorgu planı ve aday kaynaklar belirlendikten sonra aday kaynaklar iş üretme hızlarına göre sıralanmaktadır. Sıralama işlemi tamamlandıktan sonra, tüm görevler listedeki ilk aday kaynağa gönderilmektedir. Daha sonra, sorgu çalıştırılırken kaynakların başarımlarına göre kaynak ataması güncellenmektedir. Venugopal ve arkadaşları (Venugopal, et al., 2006) çalışmasında veri örgüleri üzerinde elektronik bilim (*ing. e-science*) uygulamaları planlaması için bir örgü servis aracı sunmuşlardır. Yazarlar bu çalışmanın kaynak atama bölümünde uyarlanabilir kaynak atama algoritmaları ve aracı yöntemleri üzerinde yoğunlaşmışlardır.

Görev merkezli kaynak atama yöntemleri görevlere en güçlü kaynakları atamak kaydıyla sorguların çalışma sürelerini eniyilemeyi hedeflemişlerdir. Çalışmaların çoğunda farklı görevlerin farklı özellikteki kaynak gereksinimleri, kaynakların puanlaması yapılırken her özellik için farklı katsayı kullanımı ile karşılanmıştır. İncelenen çalışmalarda durağan kaynak ataması sınıfındaki yöntemlerin çoğu, görevler arasındaki iletişim gereksinimlerini irdilemiştir. Buna karşılık devingen kaynak ataması sınıfındaki yöntemlerin çoğu görevler arasındaki iletişim gereksinimlerini irdilememiştir. Bu çalışmalar daha çok devingenlik konusuna odaklanmışlardır. İncelenen görev merkezli kaynak atama yöntemlerinde sıklıkla açgözlü algoritma yaklaşımı (*ing. greedy approach*) kullanılmıştır. Bu şekilde NP-Complete olarak nitelendirilen kaynak atama problemine yaklaşımsal (*ing. approximation*) çözümler önerilmiştir. Bu sebeple görevlere genellikle en kuvvetli kaynaklar atanmaktadır. Kaynakların puanlamasında birçok kaynak özelliği kullanılmıştır. Fakat kaynakların kullanım oranı birincil ölçüt olarak irdelenmemiştir. Bu da sistemdeki yük dağılımının dengesiz olmasına sebep olabilmektedir. Bu sınıfta incelenen çalışmaların çoğu merkezi çalışacak şekilde tasarlanmışlardır. Çalışmalarda karmaşıklığı azaltmak amacıyla çeşitli yöntemler kullanılmışsa da örgü sistemleri gibi devingen ve geniş ölçekli ortamlarda merkezi tasarımlar tekil hata noktası ve darboğaz problemine sebep olabilmektedir.

2.3.2.2 Kaynak merkezli kaynak atama algoritmaları

Kaynak merkezli kaynak atama yöntemleri kaynak atama problemine kaynaklar tarafından yaklaşmaktadır. Bu bağlamda birincil amaç kaynak kullanımını ve kaynaklar arası yük dengelemesini eniyilemektir. Burada kullanılan fikir, kaynak kullanımının azami dereceye çıkarılmasının ve yük dengesinin sağlanmasının görev yürütmesinde verimliliği artırmasıdır. Literatürde bu sınıfa ait birçok çalışma bulunmaktadır. Bu çalışmaların çoğu devingen kaynak atama algoritmaları üzerine yoğunlaşmışlardır (Gounaris, et al., 2005; Gounaris, et al., 2009; Kotowski, et al., 2008; Paton, et al., 2009; Quiane-Ruiz, et al., 2009), durağan kaynak ataması konusundaki çalışmalar daha az sayıdadır (Soe, et al., 2005). Soe ve arkadaşları (Soe, et al.) çalışmasında örgü sistemlerinde koşut sorgu işlemesi için bir kaynak atama algoritması sunmuşlardır. Kaynak atamasında sorgu içi ve sorgular arası koşutlarının herikisi de irdelenmiştir. Maliyet ölçütü olarak sadece kaynak kullanım oranları kullanılmıştır, kaynakların diğer özellikleri irdelenmemiştir. (Gounaris, et al., 2005) çalışmasında Gounaris ve arkadaşları örgü sistemlerinde sorgu işlemesi için kaynak atama algoritması

sunmuşlardır. Önerilen algoritma servis odaklı bir yöntemdir, bu sebeple özerk olarak nitelendirilmektedir. Algoritmadaki başlangıç kaynak ataması (Gounaris, et al., 2004) çalışmasında sunulduğu şekilde gerçekleştirilmektedir. Daha sonra, görevlere atanan kaynaklar üzerindeki sorgu değerlendirme servisi aracılığı ile sorgunun başarımına göre kaynak ataması güncellenmektedir. Paton ve arkadaşları (Paton, et al., 2009) çalışmasında dağıtık geniş ölçekli devingen ortamlar için bir sorgu işleme algoritması sunmuşlardır. Çalışmada yük dengelemesine dayalı sorgu eniyilemesi amaçlanmıştır. Kotowski ve arkadaşları (Kotowski, et al., 2008) çalışmasında örgü sistemlerinde sorgu içi ve sorgular arası koşut irdeleyen bir koşut sorgu işleme yordamı sunmuşlardır (GParGRES). Önerilen yöntem daha önceki çalışmaları olan ParGRES veritabanı küme ara katmanına (Mattoso, et al., 2005) dayanmaktadır. ParGRES hem sorgu koşutunu hem de yük dengelemesini içeren bir ara katman yazılımıdır ve yük dengelemesi için uyarlanabilen kaynak ataması yöntemi kullanmaktadır. Gounaris ve arkadaşları (Gounaris, et al., 2009) çalışmasında örgü sistemlerinde uyarlanabilir sorgu işleme yöntemi önermişlerdir. Çalışmada hem konum hem de veri bölümlenmesi durumları irdelenmiştir. Önerilen yöntem devingen kaynak atamasını amaçlamaktadır. Ruiz ve arkadaşları (Quian'e-Ruiz, et al.) çalışmasında dağıtık bilgi sistemleri için bir sorgu atama yöntemi sunmuşlardır. Üzerinde çalışılan ortamın türdeş olmayan ve devingen kaynaklardan oluştuğu varsayılmıştır. Önerilen algoritma memnuniyet tabanlı bir modele dayanmaktadır.

Kaynak merkezli kaynak atama yöntemleri genel olarak aday kaynaklar arasında yük dengelemesini sağlamayı amaçlamışlardır. Bu sınıfta incelenen çalışmaların hemen hepsi görevlerin ihtiyaçlarından ziyade kaynaklar arasındaki yük dağılımına odaklanmışlardır. Bu sebeple görevlere özgü farklı gereksinimler ve görevler arasındaki iletişim gereksinimleri bu sınıftaki çalışmaların çoğunda irdelenmemiştir. Öte yandan, kimi çalışmalar yük dengelemesinin yanı sıra kaynakların bazı temel özelliklerini de hesaba katmışlardır. Bu çalışmalar kaynaklar üzerindeki yük dağılımının daha hassas hesaplanabilmesine olanak sağlamaktadır. Bu sınıfta incelenen çalışmalardaki ana fikir, kaynak kullanımının arttırılması ve yük dengesinin sağlanmasının sistemin bütününe verimliliğini arttırmasıdır. Fakat görevler arasındaki iletişim gereksinimlerinin ve görevlerin farklı gereksinimlerinin göz ardı edilmesi başarımın düşmesine sebep olabilmektedir. Bu durumlarda her ne kadar kaynaklar arası yük dengesi sağlansa da görevler arasındaki iletişim trafiği sistemin başarımını olumsuz etkileyebilmektedir.

2.3.2.3 Sınıflar arasında niteliksel karşılaştırma

Tezin bu bölümünde incelenen her iki farklı sınıftaki kaynak atama yöntemleri de örgü sistemlerinde sorgu işlemesi için kaynak atamayı hedeflemektedir. Her ne kadar farklı sınıflardaki yöntemler birçok ortak özellik taşımaktaysa da aralarında bir çok farklılıklar da bulunmaktadır. Bu farklılıklar, sınıflar arasında karşılaştırma yapmaya ve sınıfların kendilerine özgü iyi ve kötü yanlarının ortaya konulmasına olanak sağlamaktadır. İncelenen iki farklı sınıftaki yöntemlere ilişkin karşılaştırma özeti çizelge 2.5'te görülebilir. İncelenen yöntemlerin hemen tümünde kaynakların özellikleri irdelenmiştir. Fakat görev merkezli kaynak atama yöntemlerinde görevler arasındaki bağımlılıklar, görevlere has farklı gereksinimler ve kaynakların donanımsal özellikleri ağırlıklı olarak incelenmiştir ve bu değiştirgelere dayalı detaylı puanlama fonksiyonları sunulmuştur. Öte yandan bu sınıftaki yöntemler örgü üzerindeki kaynaklarda yük dengesinin bozulmasına ve kimi kaynakların aşırı yüklenmesine sebep olabilmektedirler. Bu da koşut işlemlerin tamamlanma süreleri arasında farklılıklar oluşmasına neden olabilmektedir. Buna karşılık kaynak merkezli kaynak atama yöntemlerinde yük dengelemesinin sistemin bütünsel başarımını arttıracığı düşünülmektedir. Sorgu işlemesi alanında yük dengelemesi gerekli olmakla birlikte tek başına yeterli olmayabilmektedir. Bu anlamda yük dengelemesinin yanı sıra görevler arasındaki bağımlılıkların ve görevlere özgü gereksinimlerin de irdelenmesi gerekmektedir. Bu çalışmada incelenen her iki sınıf yöntemde de durağan ve devingen kaynak atama yöntemleri bulunmaktadır. Bu anlamda, durağan kaynak atama yöntemleri örgü sistemlerindeki devingenlik göz önüne alındığında devingen kaynak atama yöntemlerine göre daha az güvenilir olarak nitelendirilebilir. Devingen kaynak atama yöntemleri başlangıç kaynak atamasının ortamın özelliklerine göre gerektiği hallerde kaynak atamasını değiştirebilmektedirler. Tezin bu bölümünde incelenen kaynak atama yöntemlerinin çoğu merkezi çalışacak şekilde tasarlanmıştır. Geniş ölçekli sistemlerde merkezi tasarımlar darboğaz ve tekil hata noktası problemi oluşumuna sebep olabilmektedir. Her ne kadar çalışmaların çoğu merkezi iş yükünü azaltabilmek amacıyla çeşitli yöntemler kullanmışsalar da, bizim bilgimiz dâhilinde örgü sistemlerinin özelliklerini detaylarıyla inceleyip sorgu işlemesinde kullanılmak üzere tamamen dağıtık olarak tasarlanmış bir kaynak atama çalışması bulunmamaktadır.

Çizelge 2.5 Kaynak Atama Yöntemleri Sınıflar Arası Niteliksel Karşılaştırma Özeti

Değerlendirme Ölçütleri	Kaynak Atama Yöntemi Sınıfları	
	<i>Görev Merkezli Yöntemler</i>	<i>Kaynak Merkezli Yöntemler</i>
<i>Göreve has farklı gereksinimler</i>	Puanlama fonksiyonlarında katsayı kullanımı ile irdelenmiştir.	Çoğu çalışmada irdelenmemiştir.
<i>Görev bağımlılıkları</i>	Ağ özelliklerinin incelenmesi ve görevlerin gereksinimlerinde tanımlanarak irdelenmiştir.	Görevlerin yürütülme sırası şeklinde irdelenmiş fakat iletişim gereksinimleri şeklinde irdelenmemiştir.
<i>Yük dengesi</i>	Yük dengesi öncelikli olarak irdelenmemiştir.	Yük dengesi öncelikli olarak irdelenmiştir.
<i>Kaynakların özellikleri</i>	Detaylı puanlama fonksiyonları kullanılarak irdelenmiştir.	Çoğu çalışmada irdelenmemiştir.
<i>Ölçeklenebilirlik ve güvenilirlik</i>	Tüm çalışmalar merkezi olarak tasarlanmış, bu sebeple örgü sistemleri için ölçeklenebilirliği ve güvenilirliği düşüktür.	Merkezi olarak tasarlanan yöntemlerin örgü sistemleri için ölçeklenebilirliği ve güvenilirliği düşüktür. Dağıtık olarak tasarlananlar ölçeklenebilir ve güvenilir olarak nitelendirilebilir.

2.3.2.4 Sınıflar arasında niceliksel karşılaştırma

Bu kısımda, önerilen sınıflamada her sınıfın temel özelliklerini yansıtan basite indirgenmiş birer örnek kaynak atama uygulaması benzetim ortamında uygulanmıştır. Benzetim sonuçları da dikkate alınarak sınıflar arasında niceliksel değerlendirme ve karşılaştırmalar sunulmaktadır.

Benzetim ortamı olarak GirdSim (Sulistio, et al.) örgü benzetim yazılımı kullanılmıştır. GridSim örgü sistemi üzerinde görev planlama yöntemlerinin geliştirilebilmesi amacıyla hazırlanmış bir ayrık olay benzetimcisidir (*ing. discrete*

event simulator). GridSim’de kaynaklar işlemci hızları (MIPS), işlemci sayısı, işlemci yükü ve bağlantı hızı (baud hızı) dikkate alınarak modellenmiştir. Görevler ise girdi verisi büyüklüğü, yönerge sayısı ve çıktı verisi büyüklüğü dikkate alınarak ifade edilmiştir.

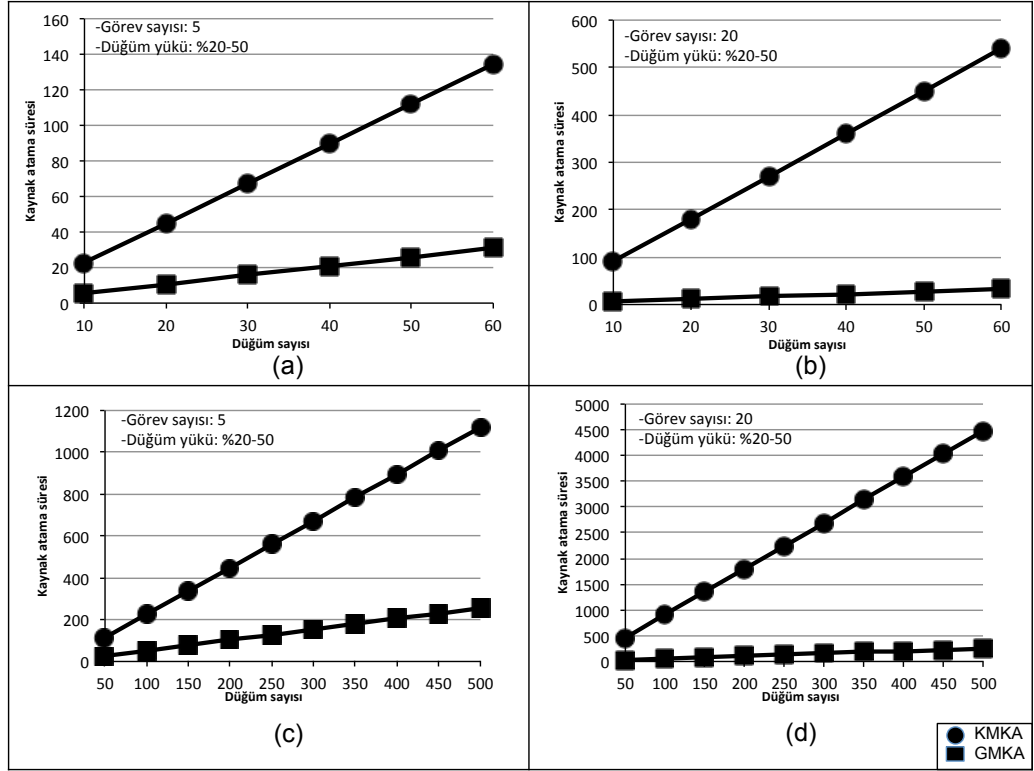
Tezin bu bölümünde gerçekleştirilen benzetimlerde görevler sorguların içerisindeki operatörleri oluşturan temel işlemler olacak şekilde irdelenmiştir. Bu anlamda basit, orta ve karmaşık sorgular sırası ile 5, 10 ve 20 adet göreve denk gelecek şekilde modellenmiştir. 20’den daha fazla sayıdaki görev, çalıştırılmayı bekleyen birden fazla sorguyu temsil etmektedir. Benzetimlerin tümünde görevlerin birbirlerinden bağımsız olduğu ve birbirleriyle iletişim kurmadıkları varsayılmıştır.

Benzetimlerde kaynaklar tek işlemcili düğümleri temsil etmektedir. Düğümler birbirlerine baud hızı ve yayılma gecikmesi (*ing. propagation delay*) irdelenen sanal iletişim kanalları ile bağlanmıştır. Aday düğümler, özellikleri belirli aralıklar içerisinde tekdüze dağılım (*ing. uniform distribution*) kullanılarak oluşturulmuşlardır. Farklı özelliklerdeki görevler de aynı şekilde rasgele oluşturulmuştur. Benzetimde gerçekleştirilen testlerde dört adet değiştirge kullanılmıştır: kaynak atama sınıfı, düğüm sayısı, görev sayısı ve düğümlerdeki işlemci yükü. Kaynak atama sınıfı olarak görev merkezli kaynak atama (GMKA) ve kaynak merkezli kaynak atama (KMKA) olmak üzere iki farklı sınıf irdelenmiştir. GMKA’da görevler büyüklüklerine göre, kaynaklar da işlemci hızlarına göre sıralanmaktadır. Daha sonra görevler kaynaklara aynı sıra ile atanmaktadır. Kaynak sayısı yeterli olmadığı takdirde kaynak listesi başa dönülerek kaynak ataması işlemi devam ettirilmektedir. KMKA’da ise her bir görev için kaynaklar arasından en az yüklü olan kaynak seçilmekte ve atama işlemi gerçekleştirilmektedir. Deneyler sırasında kaynak atama algoritmasının çalışma süresi ve görevlerin tamamlanma süresi olmak üzere iki farklı ölçüm yapılmıştır. Alınan ölçüm değerleri benzetim ortamı zaman birimindedir ve kullanılan birçok değiştirgeye bağlı olarak değişebilmektedir, bu sebeple gerçek veya gerçeğe yakın ölçümleri ifade etmeleri amaçlanmamıştır. Göreli olarak iki farklı sınıf arasında karşılaştırma yapabilmek amacıyla kullanılmıştır.

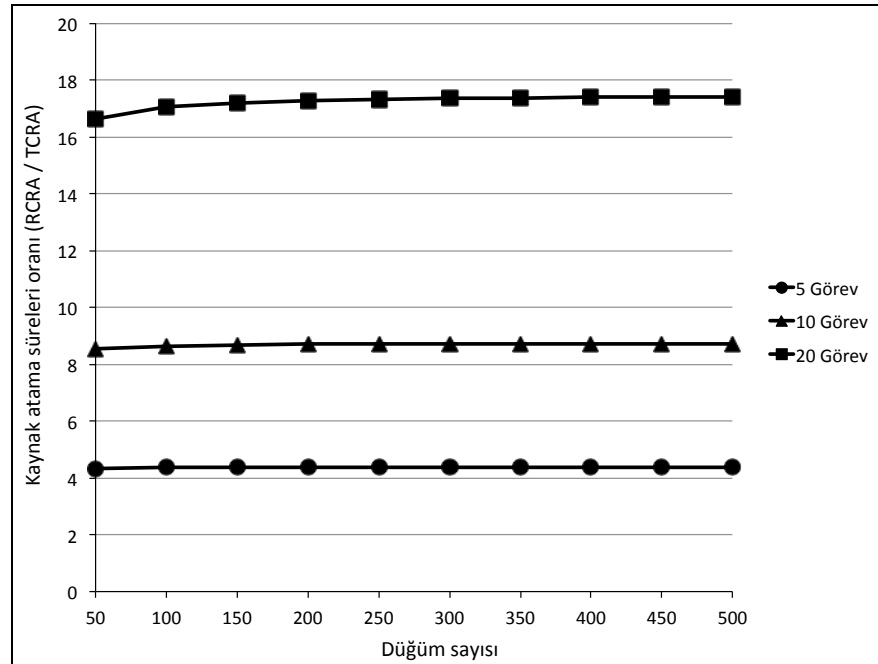
Şekil 2.3’te, düğüm sayısı 10 ila 500 arasında değişen senaryolar üzerinde kaynak atama işlemi için gereken süre ölçülmüştür. Testlerde 5 ve 20 adet görev içeren senaryolar denenmiştir. Düğümlerin işlemci yükleri %20 ile %50 arasında olacak şekilde rastgele olarak atanmıştır. 10 ila 60 düğümden oluşan küçük ölçekli

ortamda 5 ve 20 görevden oluşan testler şekil 2.3.a ve şekil 2.3.b’de görülebilir. Aynı testin 50 ila 500 düğümden oluşan geniş ölçekli ortamdaki sonuçları ise şekil 2.3.c ve şekil 2.3.d’de sunulmuştur.

GMKA algoritmasının zaman karmaşıklığı (i) puanlama için her düğümün özelliklerinin alınması, (ii) düğümlerin puanlarına göre sıralanması ve (iii) görevlerin uzunluklarına göre sıralamasından oluşmaktadır. Buna göre bu algoritmanın teorik zaman karmaşıklığı m düğüm sayısını, n görev sayısını ve c düğüm özelliklerinin alınması sırasındaki iletişim masrafı olacak şekilde $O(m * c + n * \log n + m * \log m)$ olarak hesaplanmaktadır. Öte yandan, KMKA algoritmasının zaman karmaşıklığı her bir görev için tüm düğümlerin yük durumlarının alınmasından oluşmaktadır. Buna göre KMKA algoritmasının zaman karmaşıklığı $O(n * m * c)$ olarak hesaplanmaktadır. Hazırlanan benzetim ortamında c sabiti fonksiyonlardaki diğer değişkenlere oranla çok büyük olması sebebiyle algoritmaların teorik çalışma süreleri c sabitini içeren terim ile eşleşmektedir. Benzetimlerden elde edilen test sonuçları bu analizleri doğrular niteliktedir. Şekil 2.3'te görüleceği üzere her iki algoritmanın da çalışma zamanları düğüm sayısı arttıkça artmaktadır. Fakat KMKA algoritmasının ölçümlerindeki artış GMKA algoritmasındaki artışa göre daha hızlı yükselmektedir. Daha kolay karşılaştırma yapılabilmesi açısından Şekil 2.4'te algoritmaların kaynak atama sürelerinin oranı verilmektedir. Bu grafikte gösterilen veriler KMKA algoritmasının çalışma süresinin GMKA algoritmasının çalışma süresine bölünmesiyle elde edilmiştir. Buna göre görev sayıları sabitken artan düğüm sayısına göre algoritmaların çalışma sürelerinin oranları neredeyse sabit kalmaktadır. Bunun sebebi algoritmaların kuramsal çalışma süreleri oranlarının görev sayısına bağımlı olmasıdır. Şekil 2.4'te görülebileceği üzere düğüm sayısı sabit tutulduğunda algoritmaların çalışma süresi oranı görev sayısı ile doğru orantılı olarak değişmektedir.



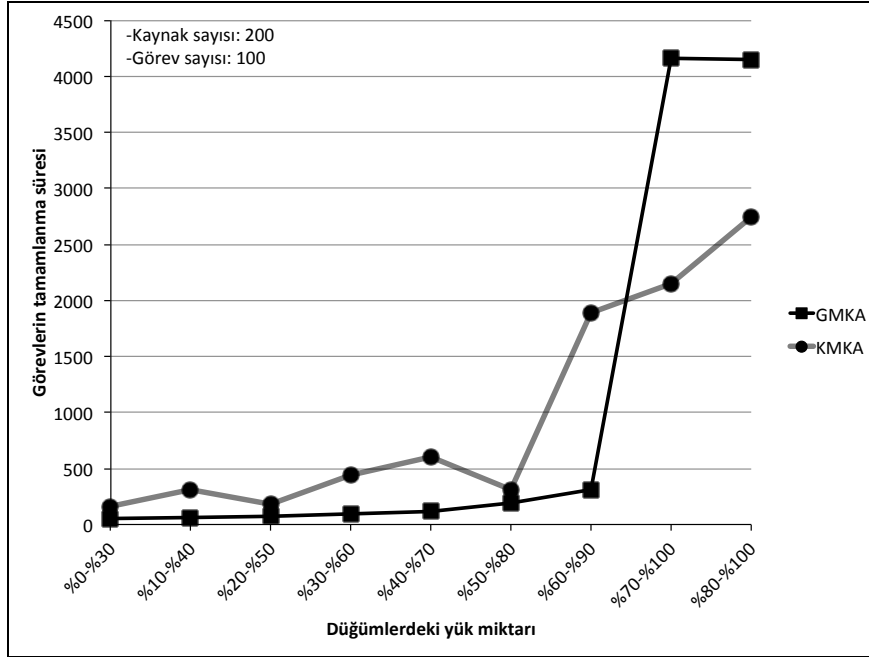
Şekil 2.3 Değişen ortam büyüklüklerine göre kaynak atama maliyetleri



Şekil 2.4 Değişen düğüm sayılarına göre farklı görev yükleri için başarımları

Şekil 2.5'te düğümlerdeki değişen yük miktarlarına göre görevlerin tamamlanma süresi ölçülmüştür. Bu test sırasında kaynak sayısı 200, görev sayısı ise 100 olarak sabitlenmiştir. Düğümlerdeki yük miktarları %0-30, %10-40, %20-

50,..., %70-100 ve %80-100 aralıklarında olacak şekilde ölçümler yapılmıştır. Bu deneyde alınan sonuçlar, düğümlerdeki yük miktarının görece olarak düşük olduğu ortamlarda GMKA algoritmasının daha iyi sonuçlar verdiğini göstermektedir. Buna karşılık, düğümlerdeki yük miktarının %70'ten daha büyük olduğu durumlarda KMKA algoritması daha iyi sonuçlar vermiştir.

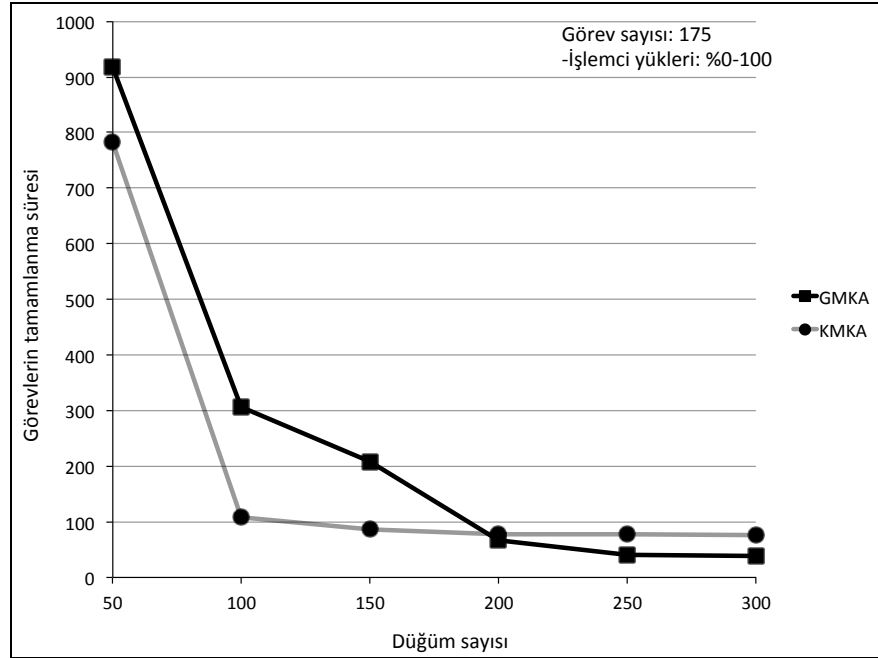


Şekil 2.5 Değişen işlemci yüklerine göre görev tamamlanma süreleri

Şekil 2.6'te görevlerin tamamlanma sürelerinin değişen düğüm sayısı karşısındaki değerleri ölçülmüştür. Bu testte düğümlerin yükleri %0 ile %100 arasında değişecek şekilde rastgele atanmıştır. Görev sayısı 175 olarak sabitlenmiştir. Bu testte amaç iki farklı ortam durumunda görevlerin tamamlanma sürelerinin gözlemlenmesidir: (D1) düğüm sayısı görev sayısından az iken, (D2) düğüm sayısı görev sayısından fazla iken. Şekilde de görülebileceği üzere D1 durumunda KMKA algoritması daha iyi sonuçlar vermektedir. Bunun sebebi düğüm sayısının görev sayısından küçük olması halinde yük dengelemesinin daha fazla önem kazanmasıdır (Pourebrahimi and Bertels, 2007). Öte yandan, D2 durumunda GMKA algoritması daha iyi sonuçlar vermiştir. Bunun sebebi D2 durumunda seçilen düğümlerdeki yük miktarlarının düşük olma ihtimalinin daha yüksek olmasıdır.

Benzetim sonuçları göz önüne alındığında, düğümlerin yüklü olduğu ve kaynak sayısının görev sayısına göre daha az olduğu durumlarda KMKA algoritmasının daha iyi sonuçlar verdiği söylenebilir. Fakat bu durumlarda KMKA

algoritmasının zaman karmaşıklığının da göz önüne alınması gerekmektedir. Öte yandan, düğüm sayısının görev sayısına göre daha fazla olduğu ve düğümlerinin ortalama yüklerinin belirli bir seviyeden düşük olduğu durumlarda GMKA algoritması daha iyi sonuçlar vermiştir.



Şekil 2.6 Değişen düğüm sayılarına göre görevlerin tamamlanma süreleri

2.3.3 Örgü sistemlerde sorgu işlemede hataya dayanıklılık

Örgü sistemleri devingen olmaları sebebiyle düğümlerin sistemden habersiz ayrılmaları olağan bir durum olarak kabul edilmektedir. Bu gibi düğüm hataları sorgunun çalışmasına katılan düğümlerde gerçekleştiğinde sorgunun yeni baştan çalıştırılması yüksek maliyetli olabilmektedir. Bu sebeple örgü sistemlerde sorgu işleminde hataya dayanıklılık çok önemli bir konu olarak irdelenmektedir.

Literatürde devingenliği irdemiş birçok kaynak atama çalışması bulunmaktadır (Gounaris, et al., 2005; Silva, et al., 2006; Venugopal, et al., 2006; Kotowski, et al., 2008; Gounaris, et al., 2009; Paton, et al., 2009; Quiane-Ruiz, et al., 2009). Ancak bu çalışmalar düğümlerin başarımlarının devingenliğini irdemiş, durumlu işletmenlerin (*ing. Stateful operators*) çalışması sırasında gerçekleşen düğüm hatalarını ve düğümlerin sistemden ayrılmalarını irdedememiştir. Kodlayarak birleştirme (*ing. Hash join*) gibi durumlu işletmenlerin durumlarının düğüm hatası sonucu kurtarılması gerektiğinden yukarıdaki devingen kaynak atama algoritmaları bu gibi durumlarda uygun

olmayabilir. Bilgimiz dâhilinde örgü sistemler üzerinde hataya dayanıklı sorgu işlemesi sunan az sayıda çalışma bulunmaktadır (Smith and Watson, 2005; Smith and Watson, 2007; Taylor, 2008; Bestehorn, et al., 2010). Her ne kadar bu çalışmalar başarılı çözümler sunsalar da çalışmalardan hiçbirisi durumlu işletmenlerin çalıştırılması üzerinde özelleşmemiştir.

2.3.3.1 Yakın zamanda yapılan çalışmalar

Literatürde örgü sistemlerde sorgu işlemesi için kaynak ataması üzerine birçok çalışma bulunmaktadır (Gounaris, et al., 2004; Gounaris, et al., 2005; Kant and Grosu, 2005; Mandal, et al., 2005; Soe, et al., 2005; Gounaris, et al., 2006; Silva, et al., 2006; Venugopal, et al., 2006; Bose, et al., 2007; Kotowski, et al., 2008; Liu and Karimi, 2008; Gounaris, et al., 2009; Paton, et al., 2009; Quiane-Ruiz, et al., 2009). Bu çalışmalar, kaynak atamasının bir defaya mahsus yapıldığı durağan (*ing. static*) kaynak ataması veya kaynak atamasının sorgu işlemesinin başarımına göre yinelenildiği devingen (*ing. dynamic*) kaynak ataması önermektedirler. Her ne kadar devingen kaynak atama algoritmaları ortamın devingenliğini hesaba katsalar da, düğümlerin sistemden ayrılması durumunu irdelememektedir. Bu durumu inceleyen çalışmalar bulunmakla beraber (Abadi, et al., 2005; Balazinska, et al., 2008; Bestehorn, et al., 2010; Chandrasekaran, et al., 2003; Hwang, et al., 2005; Jeong-hyon Hwang, et al., 2007; Kwon, et al., 2008; Smith and Watson, 2005; Smith and Watson, 2007) bunlardan pek azı birleştirme işletmeni gibi durumlu işletmenlerin çalıştırılması üzerinde yoğunlaşmıştır (Bestehorn, et al., 2010).

Smith ve Watson (Smith and Watson, 2005) çalışmasında dağıtık sorgu işlemesi için hataya dayanıklı bir sorgu işleme sistemi sunmuşlardır. Bu çalışmaları daha önceden geliştirdikleri *OGSA-DQP* (Alpdemir, et al., 2004) sistemine ek olarak hazırlanmıştır ve hata algılayıcı, hata çözücü gibi hataya dayanıklılık bileşenleri içermektedir. Çalışmalarında sorgunun gönderildiği düğüm bir sorgu işletme planı oluşturmakta, başlangıç kaynak ataması gerçekleştirmekte ve sorgunun işleyişini başlatmaktadır. Hataya dayanıklılık için geriye dönüş protokolü kullanılmaktadır. (Smith and Watson, 2007) çalışmasında ise, Smith ve Watson örgü sistemlerde sorgu işleme için hatadan düzeltme yöntemlerini tartışmışlardır. Çalışmalarında yeniden başlatma, seyreltme ve yerine koyma olarak adlandırılan farklı hata düzeltme yöntemlerini incelemiştirler. Bestehorn ve arkadaşları (Bestehorn, et al., 2010) çalışmasında yapılandırılmış eşler arası sistemler için hataya dayanıklı bir sorgu işleme sistemi sunmuşlardır.

Çalışmalarında sorgu işletmenlerini konumlu ve konumsuz olacak şekilde iki sınıf altında incelemiştir. Sunulan yöntem her iki sınıftaki işletmenler için hataya dayanıklılık sağlamaktadır. Sunulan çalışma hata saptaması ve yedek kaynak seçimi için CAN eşler arası sisteminin sunduğu servislerden faydalanmaktadır. Hataya dayanıklı sorgu işleme sistemleri hakkında geniş ve detaylı bir literatür özeti (Taylor, 2008) çalışmasında sunulmuştur. Taylor bu çalışmada hataya dayanıklı dağıtık sorgu işleme yöntemlerini 3 sınıf altında incelemiştir: *i)* yukarıya doğru yedekleme (Smith and Watson, 2007; Smith and Watson, 2005), *ii)* etkin bekleme (Abadi, et al., 2005; Balazinska, et al., 2008; Chandrasekaran, et al., 2003) ve *iii)* edilgin bekleme (Hwang, et al., 2005; Jeong-hyon Hwang, et al., 2007; Kwon, et al., 2008).

2.3.3.2 Sonuçlar

Hataya dayanıklı sorgu işleme konusu üzerinde yapılan değerlendirmeler dikkate alındığında bu konu ile ilgili literatürde bir çok çalışma yapıldığı görülmektedir. Her ne kadar bu çalışmalar etkin çözümler sunmuş olsalar da bilginiz dâhilinde konumlu işletmenlerin örgü sistemleri üzerinde hataya dayanıklı işletilmesi konusunda özelleşmiş çalışmalara rastlanılmamıştır. Bu konuda incelenen çalışmaların büyük bölümü konulu olmayan görevlerden oluşan akış işleme konusu üzerinde özelleşmiştir. Diğer çalışmalar ise eşler arası sistemler üzerinde çalışacak şekilde geliştirilmiştir. Bu sebeple örgü sistemlerinin özellikleri ve konumlu sorgu işletmenlerinin gereksinimleri gözetilerek örgü sistemleri üzerinde sorgu işlemesi için hataya dayanıklı kaynak atama algoritmalarının önerilmesi konusunda eksiklikler olduğu düşünülmektedir.

2.3.4 Kaynak atama algoritmaları için sonuçlar

Tezin bu bölümünde örgü sistemleri üzerinde sorgu işlemesi için kaynak ataması yöntemleri sınıflandırılarak incelenmiş, niceliksel ve niteliksel ölçütlere göre değerlendirilmiş ve sınıflar arası karşılaştırmalar yapılmıştır. Bunun yanı sıra her iki sınıfın temel özelliklerini yansıtacak şekilde iki kaynak atama algoritması GridSim benzetim ortamında denenmiş ve deney sonuçlarına göre yorumlamalar ve karşılaştırmalar sunulmuştur. Bu anlamda mevcut kaynak atama yöntemlerinin faydaları ve eksiklikleri ortaya konulmuştur. Elde edilen sonuçların yeni yöntemlerin önerilmesinde yol gösterici olması amaçlanmıştır. Buna ek olarak hataya dayanıklı kaynak atama algoritmaları da incelenmiş, literatürdeki çalışmalar ile ilgili değerlendirmeler sunulmuştur.

Her ne kadar mevcut kaynak atama algoritmaları bu soruna başarılı çözümler üretmiş olsalar da, örgü sistemlerin özellikleri ve sorguların gereksinimlerinin tümünü karşılayan çalışmalar bilginiz dâhilinde bulunmamaktadır. Bu kısımda incelenen çalışmaların kendine has fayda kazanımları ve fayda yitimleri bulunmaktadır. Bu sebeple bu aşamada incelenen sınıflar arasında üstün olanın seçilmesi mümkün görülmemektedir. Bunun yerine mevcut algoritmalar değiştirgesel sorgu eniyilemesi gibi sistemlerde (Bizarro, et al., 2009; Ioannidis, et al., 1997) duruma uygun olarak kullanılacak farklı seçenek olarak sunulabilmektedirler.

2.4 Genel Sonuçlar

Bu bölümde, örgü sistemlerde sorgu işleme için kaynak tarama, ilinge denetimi için öz-kararlı kapsayan ağaç oluşturma, kaynak atama ve hataya dayanıklı kaynak atama algoritmaları ile ilgili detaylı bir literatür taraması çalışması sunulmuştur. Mevcut algoritmalar sınıflandırılarak incelenmiş ve önceden belirlenen değerlendirme ölçütlerine göre değerlendirilmiştir. Sınıflar arası karşılaştırmalar sunulmuş ve her bir kısım için sonuçlar ortaya konulmuştur. Gerçekleştirilen değerlendirmeler dikkate alındığında incelenen problemlerin her birinde halen açık noktalar ve olası katkı konulabilecek eksikler olduğu görülmektedir. Her bir problem için ilgi çeken açık noktalar ve olası katkı konuları aşağıda listelenmiştir:

i. Örgü sistemlerinde sorgu işleme için kaynak taraması: Örgü sistemlerinin devingen olması ve kaynak özelliklerinin zaman içinde değişebilmesi sebebiyle, güncel kaynak bilgilerine ulaşılabilmesi için kaynak taraması sorgularının örgüdeki her bir düğüm tarafından yanıtlanması gerektiği düşünülmektedir. Bu bağlamda yapılandırılmamış eşler arası sistemler tabanlı kaynak tarama yöntemlerinin uygun olduğu düşünülmektedir. Ancak örgü sistemlerin geniş ölçeklilik özelliği dikkate alındığında bu yöntemlerde kullanılan taşıma işlemi KT algoritmasının ölçeklenebilirliğini kısıtlamaktadır. Dağıtık sistemlerde taşıma şeklindeki mesaj dağılımına alternatif olarak birçok verimli ilinge denetimi bulunmaktadır. Bu sebeple, kaynak tarama sorgularının dağıtımı için kaynak bilgilerini merkezileştirilmeden bir ilinge denetimi kullanılması soruna büyük ölçüde çözüm getirmektedir. Bu şekilde *i)* mesaj ve zaman karmaşıklığını azaltmayı, *ii)* güncel kaynak bilgilerine ulaşmayı ve *iii)* ilingedeki değişikliklere ve düğümlerin devingenliğine dayanıklı bir yöntem geliştirmeyi amaçlayan algoritmalar geliştirilmelidir. Bu bağlamda, öz-kararlı kapsayan ağaç

ilingesinin ve onun üzerinde verimli bir mesaj iletim yönteminin bu probleme verimli bir çözüm üretmesi düşünülmektedir.

ii. Öz-kararlı kapsayan ağaç oluşturma: Bu bölümde incelenmiş öz-kararlı kapsayan ağaç oluşturma algoritmaları bu probleme kapsamlı çözümler üretmişlerdir. Ancak kapsayan ağaç oluşturulurken tüme gönderim ve tümden toplama işlemlerini verimli hale getirecek kapsayan ağaç oluşturmaya odaklı bir çalışma, bilğimiz dâhilinde bulunamamıştır. Örneğin oluşacak kapsayan ağacın çapının küçültülmesi bu işlemlerin karmaşıklığını azaltacaktır. Bu sebeple küçük çaplı kapsayan ağaç oluşturacak öz-kararlı algoritmaların önerilmesinin bu alanda bir katkı sağlayacağı düşünülmektedir.

iii. Örgü sistemlerde sorgu işleme için kaynak ataması: Örgü sistemlerde sorgu işleme için kaynak ataması problem literatürde çokça irdelenmiş bir konudur. Ancak, örgü sistemlerinin kıyasla yeni bir kavram olmaları, mevcut çalışmaların genelde koştur ve dağıtık sistemler için hazırlanmış algoritmaların uzantısı olduğu görülmüştür. Özellikle örgü sistemleri üzerinde sorgu işlemede kullanılacak şekilde tasarlanmış az sayıda çalışmaya rastlanmıştır. Bu sebeple, kaynak atama problem üzerine katkı koyulabilecek birçok açık nokta olduğu tespit edilmiştir. Örneğin aday düğüm örnek uzayının büyüklüğü hiçbir çalışmada dikkate alınmamıştır. Örgü sistemler gibi geniş ölçekli sistemlerde aday düğüm örnek uzayı aday düğümlerin tümünün değerlendirilmesine engel olabilecek boyutlarda olabilmektedir. Bu sebeple aday düğümler incelenirken örnek uzayın küçültülmesi gerekmektedir. Bunun yanı sıra mevcut kaynak ataması çalışmaları içerisinde çok azı kaynak seçiminde iletişim maliyetleri üzerine odaklanmıştır (Bose, et al., 2007; Liu and Karimi, 2008). Özellikle sorgu işleminde düğümler arasında aktarılan verinin büyüklüğü göz önüne alındığında iletişim maliyetlerinin öncelikli olarak irdelenmesi gerekmektedir. Bütün bunlar göz önüne alındığında, örgü sistemler üzerinde sorgu işleme için kaynak ataması konusunda halen açık konuların bulunduğu düşünülmektedir.

iv. Hataya dayanıklı kaynak atama algoritmaları: Örgü sistemleri doğaları gereği devingen ortamlardır. Mevcut çalışmalarda devingenlik iki farklı anlamda kullanılabilir: *i)* kaynakların özelliklerinin devingenliği ve *ii)* örgüdeki düğümlerin ortamda mevcudiyetlerinin devingenliği. Bu konuda hazırlanan literatür taraması dikkate alındığında devingen kaynak atama algoritmalarının büyük bölümünün devingenliği ilk anlamında *(i)* kullandığı anlaşılmıştır. Ancak, devingenliğin ikinci anlamının *(ii)* sorgu işleminin

başarımında büyük etkisi olacağı düşünülmektedir. Bu konuda, devingenliği ikinci anlamı (ii) ile kullanan az sayıda çalışma bulunmuştur. Bu sebeple hataya dayanıklılığın kaynak atama konusuna büyük katkı sağlayacağı düşünülmektedir.

3. ÖRGÜ SİSTEMLERDE SORGU İŞLEME İÇİN KAYNAK TARAMASI

Bu bölümde iki katmandan oluşan bir kaynak tarama algoritması önerilmektedir: i) örgü sistemlerde kaynak taraması için ilinge denetimi ve ii) kapsayan ağaç tabanlı kaynak tarama algoritması. İlk olarak ilinge denetiminde kullanılmak üzere üç adet öz-kararlı kapsayan ağaç oluşturma algoritması önerilmektedir. Daha sonra kapsayan ağaç yapısını kullanan bir kaynak tarama algoritması önerilmektedir.

3.1 Giriş

Örgü üzerinde kaynak taraması (KT), işlem ihtiyaçlarının ve ortam değişkenlerinin açıkça belirtildiği bir ortamda işlemi yürütebilme yetisine sahip olan kaynakların örgü içerisinde araştırılması olarak tanımlanmaktadır. Bununla birlikte KT geniş ölçekli, devingen ve karışık özelliklere sahip bir ortamda kabul edilebilir bir süre içerisinde tamamlanmalıdır. Örgü sistemlerinde kaynak ifadesi birçok kavramı temsil edebilmektedir. Kaynak ifadesi, işlemci, bellek, depolama ünitesi, ağ gibi bir hesaplama kaynağı olabileceği gibi; önbilgi ve içeriği sağlayan bir veri kaynağı veya önceden belirlenmiş bir görevi yerine getirebilecek bir hizmeti temsil ediyor olabilmektedir. Bu kaynakların ve özelliklerinin örgü sisteminde keşfedilmesi sorgu işleminde kullanılacak kaynakların seçiminde doğrudan etkili olmaktadır. Fakat örgü sistemlerinin devingenlik, türdeş olmayan kaynaklara sahip olma ve geniş ölçeklilik özellikleri kaynak tarama işlemini zaman alıcı bir hale getirmektedir. Bu konuda yapılmış mevcut çalışmalar bölüm 2’de detaylı olarak incelenmiştir. Burada varılan sonuçlar dikkate alındığında her bir yöntemin kendine has fayda kazanımlarının ve fayda yitimlerinin olduğu gözlemlenmektedir. Bu bölümde mevcut çalışmaların fayda kazanımlarını bir araya getirmeyi amaçlayan çok katmanlı bir kaynak tarama algoritması önerilmektedir.

Mevcut çalışmalar ile ilgili yapılan literatür taramasında kaynak taraması yöntemleri üç sınıf altında incelenmiştir: merkezi-sıradüzensel sistemler tabanlı yöntemler, eşler arası sistemler tabanlı yöntemler ve etmen sistemleri tabanlı yöntemler. Her bir sınıf için yöntemler incelenmiş, ardından detaylı çözümlemeler verilmiş ve sınıflar arası karşılaştırılmalar yapılmıştır. Bu bölümde, literatür

taramasında yapılan değerlendirmeler dikkate alınarak bir kaynak tarama algoritması önerilmektedir. İlk olarak örgü sistemi üzerinde kaynak taramasında kullanılacak ilinge denetimi algoritmaları önerilmektedir. Kaynak taramada kullanılacak mesajların dağıtılmasını kolaylaştırmak amacıyla kapsayan ağaç yapısı oluşturulmaktadır. Örgü sisteminin devingenliği göz önüne alınarak kapsayan ağaç oluşturma algoritmaları öz-kararlı olacak şekilde önerilmektedir. Daha sonra kapsayan ağaç yapısından faydalanacak bir kaynak tarama algoritması önerilmektedir. Bu algoritma da kaynak tarama mesajlarının tüme gönderim ve tümünden toplama yöntemlerini kullanarak kapsayan ağaç üzerinde dağıtılması esasına dayanmaktadır.

Bu bölümün bilime katkısı iki yönlüdür:

i. Üç adet yeni öz-kararlı kapsayan ağaç oluşturma algoritması önerilmektedir. Bu algoritmalar oluşan kapsayan ağacın çapının küçültülmesini hedeflemektedir. Önerilen algoritmalar hem kendi aralarında, hem de literatürde bulunan benzer algoritmalarla karşılaştırılmaktadır.

ii. Kaynak taramada kullanılacak mesajların dağıtılmasını ve toplanmasını kolaylaştırıp karmaşıklığı azaltan kapsayan ağaç tabanlı bir kaynak tarama algoritması önerilmektedir.

Bu bölümün geri kalanı şu şekilde yapılanmaktadır: kısım 3.2’de üç adet yeni kapsayan ağaç algoritması önerilmektedir. Algoritmalar detaylı olarak açıklanmakta ve analizleri sunulmaktadır. Algoritmalar değerlendirilmekte ve benzetim ortamı sonuçları ile hem kendi aralarında hem de mevcut algoritmalar ile kıyaslamaları verilmektedir. Kısım 3.3’te kapsayan ağaç tabanlı kaynak tarama algoritması önerilmektedir. Algoritma detaylı olarak anlatılmakta, analizleri sunulmakta ve benzetim ortamı sonuçları ile değerlendirmeleri ve benzer mevcut çalışmalar ile kıyaslamaları verilmektedir. Son olarak kısım 3.4’te sonuçlar sunulmaktadır.

3.2 Örgü Sistemlerde Kaynak Tarama İçin İlinge Denetimi

Kapsayan ağaç algoritmaları dağıtık uygulamalarda sıkça kullanılan ilinge yapılarıdır. Bir kapsayan ağaç içinde bulunduğu çizgedeki tüm düğümleri içeren ve içerisinde döngü bulunmayan ağaç yapısı olarak tanımlanmaktadır. Örgü

sistemleri gibi dağıtık sistemlerin boyutlarındaki büyüme ile birlikte kapsayan ağaç gibi ilinge denetimi sistemleri büyük ölçüde önem kazanmıştır. Bu yöntemler, dağıtık uygulamaların altında yatan çizgedeki karmaşık bağlantı yapısı sebebiyle ortaya çıkan karmaşıklığı azaltmak amacıyla kullanılmaktadır. Kapsayan ağaçların kullanımı birçok dağıtık algoritmanın etkin biçimde çalışabilmesine olanak sağlamaktadır. Özellikle tüme gönderim işlemlerini içeren algoritmalarda kapsayan ağaçlar sıklıkla kullanılmaktadır. Son zamanlarda ölçeğin de genişlemesiyle birlikte tüm sistemin bilgisinin tek bir yerde tutulmasının zor olması sebebiyle dağıtık kapsayan ağaç oluşturma algoritmaları önem kazanmıştır. Bu alanda birçok farklı tipte kapsayan ağaç oluşturma algoritması geliştirilmiştir. Birçok çalışma en küçük kapsayan ağaç bulma üzerine yoğunlaşmıştır (Ahuja and Zhu, 1989; Awerbuch, 1987; Gallagher, et al., 1983; Lien, 1988). En küçük kapsayan ağaç yapısı, bir çizge üzerinde oluşturulabilecek kapsayan ağaçlar içerisinde kenar ağırlıkları toplamı en az olan kapsayan ağaç olarak tanımlanmaktadır. En küçük kapsayan ağaçlar iletişim maliyetinin azaltılması için oldukça kullanışlı yapılardır. Bunun yanı sıra en küçük dereceli kapsayan ağaç oluşturma (Blin, et al., 2009), en küçük çaplı kapsayan ağaç oluşturma (Butelle, et al., 1995) gibi farklı algoritmalar da değişik amaçlara yönelik olarak dikkat çekmektedir.

Geleneksel dağıtık kapsayan ağaç oluşturma algoritmalarının yanında birçok çalışma devingenlik ve aksaklığa dayanıklılığı da irdelemiştir. Bu kapsamda, öz kararlı kapsayan ağaç oluşturma algoritmaları geliştirilmiştir (Gartner, 2003). Öz kararlılık kavramı ağdaki devingenliklere rağmen kapsayan ağaç yapısının yeni baştan oluşturulmaya gerek kalmadan geçerliliğini koruması esasına dayanmaktadır. Daha genel olarak açıklamak gerekirse, öz kararlılık dağıtık sistemlerin aksaklıklar halinde sonlu zaman içerisinde istenen konuma geri dönebilmesi olarak tanımlanmıştır. Bu kavram ilk olarak 1974 yılında Dijkstra (Dijkstra, 1974) tarafından ortaya atılmıştır. Öz kararlı algoritmalarda her düğüm yerel değişkenler tutmaktadır. Düğümler bu değişkenlere ve komşularının değişkenlerine göre hareket etmektedir. Düğümlerin yerel değişkenleri yerel konumlarını, tüm düğümlerin yerel konumları ise sistemin konumunu tanımlamaktadır. Hedef, sistemin konumunu geçerli konumda tutmaktır. Öz kararlılık, devingenliğin yüksek olduğu örgü sistemleri gibi ortamlarda birçok avantaj sağlamaktadır, bu sebeple örgü üzerinde kapsayan ağaç oluşturma algoritmalarında kullanılmaları birçok durumda uygundur.

Bu kısımda, düğümlerin komşuluk derecelerini dikkate alan yeni öz-kararlı kapsayan ağaç oluşturma algoritmaları önerilmektedir. Düğümlerin komşuluk dereceleri, en yüksek dereceye sahip düğümün kök olarak seçilebilmesi konusunda yardımcı olacaktır. Bu sezgisel yaklaşım, ağaç üzerindeki kök düğümün diğer düğümlere oranla daha fazla iletişim kurması gözlemine dayanmaktadır. Derecesi yüksek düğümün kök olarak seçilmesinin hem oluşacak kapsayan ağacın çapının küçültülmesine hem de daha yüksek dereceli düğümlerin kök ve köke yakın olacak şekilde konumlanmasına olanak vereceği düşünülmektedir. Bu şekilde kapsayan ağacın daha etkin kullanılması amaçlanmaktadır. Bu kısımda önerilen algoritmaların mevcut algoritmalar ile karşılaştırılabilmesi için benzer özelliklerdeki iki mevcut algoritma da detaylı olarak anlatılmaktadır. Bu amaçla klasik kapsayan ağaç oluşturma algoritması (Kshemkalyani and Singhal, 2008) ve öz-kararlı kapsayan ağaç oluşturma algoritması (Afek, et al., 1991) irdelenmektedir. Mevcut olan ve bu tezde önerilen kapsayan ağaç oluşturma algoritmaları ns2 benzetim ortamında uygulanmakta ve karşılaştırmalar yapılmaktadır.

Bu kısmın geri kalanında, kısım 3.2.1’de seçilmiş iki mevcut kapsayan ağaç oluşturma algoritması anlatılmaktadır. Kısım 3.2.2’de ise önerilen üç öz-kararlı kapsayan ağaç oluşturma algoritması detaylı olarak anlatılmaktadır. Algoritmaların karmaşıklık analizleri kısım 3.2.3’te sunulmaktadır. Kısım 3.2.4’te benzetim ortamı uygulamaları anlatılmaktadır. Son olarak kısım 3.2.5’te sonuçlar, karşılaştırmalar, fayda kazanımları ve fayda yitimleri anlatılmaktadır.

3.2.1 Karşılaştırma amaçlı seçilmiş algoritmalar

Bu bölümde mevcut iki kapsayan ağaç oluşturma algoritması anlatılmaktadır. İlk olarak mevcut iki kapsayan ağaç oluşturma algoritması incelenmektedir: bellek açısından verimli öz kararlı kapsayan ağaç oluşturma algoritması (Afek, et al., 1991) ve zamanuyumsuz koşut zamanlı başlatıcılı kapsayan ağaç algoritması (Kshemkalyani and Singhal, 2008). Bu algoritmalar, önerilen algoritmalar ile karşılaştırma amacıyla kullanılacaktır.

3.2.1.1 Bellek açısından verimli öz kararlı kapsayan ağaç algoritması (BVKA)

Afek ve arkadaşları (Afek, et al., 1991) çalışmasında öz kararlı bir kapsayan ağaç oluşturma algoritması geliştirmişlerdir. Bu çalışmada düğümlerin eşsiz

kimlik numaraları olduğu ve her düğümün kendi komşuluğundan ve komşularının konumlarından haberdar olduğu varsayılmaktadır, bir diğer deyişle komşular hatalı çalışan bir düğümü fark edebilecek bir servisle donatılmışlardır. Bu tasarımda önceden belirlenmiş bir başlatıcı düğüm bulunmamaktadır. Her düğüm aynı algoritmayı birbirinden bağımsız olarak çalıştırmaktadır. Başlangıçta her düğüm kendisi kök olacak şekilde bir kapsayan ağaç oluşturmaktadır. Daha sonra düğümler düzenli olarak komşularının bağlı olduğu kapsayan ağaçların köklerini kontrol etmektedir. Komşuları arasında kendi kökünden daha büyük köke sahip bir komşu bulunması halinde komşuluğu içerisindeki en büyük kimlikli köke mesaj göndermekte ve olumlu yanıtın gelmesinin ardından kökünü ve ebeveynini belirlemektedir. Algoritmanın sonunda en büyük kimlikli düğüm kök olacak şekilde bir kapsayan ağaç oluşturulmaktadır. Her i düğümü komşuluğunu (N_i), ebeveynini (P_i), kökünü (R_i) ve kök düğüme olan uzaklığını (D_i) belirten değişkenlere sahiptir. Sistemin genel konumunun doğruluğu durumunda en büyük kimlikli düğüm kapsayan ağacın kökü olmaktadır. Bunun yanı sıra her düğümün uzaklık değeri ebeveyninin uzaklığından bir fazladır ($D_i = D_{\text{ebeveyn}} + 1$). Kök düğümün uzaklık değeri sıfırdır ve ebeveyni olarak kendini işaret etmektedir. Öz kararlılık için düğümler düzenli olarak kendi köklerini komşuluklarındaki köklerle karşılaştırmaktadır. Eğer herhangi bir düğüm, komşuluğunda daha büyük bir köke sahip bir komşuyla karşılaşır en büyük köke *request* tipinde bir mesaj gönderir. Bu tipte mesaj alan bir kök düğüm *grant* tipinde bir mesajla karşılık verir, bu mesaj düğümün kapsayan ağaca katılmasını sağlar. Algoritmada her i düğümü düzenli olarak aşağıdaki iki koşulu kontrol etmektedir:

$$A: [(R_i = i) \wedge (P_i = i) \wedge (D_i = 0)] \vee [(R_i > i) \wedge (P_i \in N_i) \wedge (R_i = R_{\text{ebeveyn}}) \wedge (D_i = D_{\text{ebeveyn}} + 1 > 0)]$$

$$B: A \wedge (R_i \geq \max(R_{N_i}))$$

A koşulu düğümün bir kök düğümü olduğunu veya normal bir düğüm olup kökünün kendi kimliğinden büyük ve ebeveyninin kökü ile aynı olduğunu, düğümün ebeveyni ile komşu olduğunu ve köke olan uzaklığının ebeveyninin uzaklığından bir fazla olduğunu vurgulamaktadır. B koşulu ise A koşulunun yanı sıra düğümün kökünün komşularının kökleri arasında en büyüğü olduğunu ifade etmektedir. Her iki koşulun doğru olması durumunda düğümün geçerli bir konumda olduğu kabul edilmektedir. Her iki koşulun yanlış olması durumu ise düğümün geçersiz bir konumda olduğunu göstermektedir, bu durumda düğüm kendisini kök düğüm ilan ederek algoritmayı çalıştırmaya devam eder. A

koşulunun doğru olup B koşulunun yanlış olması halinde ise düğüm en büyük köke sahip olan ağaca katılmak üzere kök düğüme mesaj gönderir ve yanıtın gelmesinin ardından kapsayan ağaca katılır. Algoritmada düğümler mesajların iletilmesi için de görev yapmaktadırlar.

3.2.1.2 Zamanuyumsuz koşut zamanlı başlatıcılı kapsayan ağaç algoritması (ZKKA)

Zamanuyumsuz koşut zamanlı kapsayan ağaç oluşturma algoritması (Kshemkalyani and Singhal, 2008) düğümlerin yalnız komşularının bilgilerine ihtiyaç duydukları klasik bir dağıtık kapsayan ağaç oluşturma algoritmasıdır. Bu algoritmada öz kararlılık özelliği bulunmamaktadır. Başlangıçta her düğüm kendisi kök olacak şekilde bir kapsayan ağaç oluşturur ve bu bilgiyi komşularına *query* tipinde bir mesajla iletir. *Query* tipinde bir mesaj alan düğüm gönderici ile kendi kökünün kimliklerini karşılaştırır, göndericinin kimliği daha büyükse göndericiyi kök olarak belirler. Bu durumda eğer düğüm bir yaprak düğümse *accept* tipinde bir mesaj gönderir, ağacın içinde bir düğüm ise bu durumu bir *query* mesajı ile komşularına iletir. Eğer göndericinin kimliği mevcut kökten küçükse düğüm *reject* tipinde bir mesaj ile yeni kök düğümünü reddeder. Protokolü başlatan düğümler tüm komşularından yanıt topladığında algoritmayı tamamlar. Düğüm başlatıcı düğüm değil ise tüm komşularından yanıt mesajlarını topladıktan sonra ebeveynine yanıt mesajı gönderir ve algoritmayı tamamlar. Algoritmaya ait detaylar (Kshemkalyani and Singhal, 2008) çalışmasında sunulmaktadır.

3.2.2 Önerilen algoritmalar

Bu bölümde üç farklı öz kararlı kapsayan ağaç oluşturma algoritması önerilmektedir. Üç algoritmada da, tasarım modeli düğümlerin dereceleri göz önünde bulundurulacak şekilde oluşturulmuştur. Buradaki amaç tüme gönderim için kullanılacak kapsayan ağacın etkinliğinin artırılması için ağacın çapını düşürmektir. Sezgisel olarak derecesi en yüksek olan düğümün kök olarak seçilmesinin kapsayan ağacın çapını azaltacağı düşünülmektedir. Bu sebeple önerilen algoritmalarından ilk ikisi en büyük dereceli düğümü kök seçecek şekilde tasarlanmıştır. Üçüncü algoritmada ise oluşturulan ağacın merkez düğümü bulunup, ağacı merkez düğüm kök olacak şekilde yeniden düzenleyen bir algoritma tasarlanmıştır. Bunun sağlanabilmesi için algoritmalarda her düğüme bir etiket atanmaktadır ve kök seçimini bu etikete göre gerçekleştirilmektedir.

Algoritmelerde düğümler komşularının konumlarını ve değişkenlerini düzenli olarak kaydetmektedir. Düğümler arasında güvenilir bir iletişim kanalının bulunduğu varsayılmaktadır. Algoritmaların üçü de aşağıdaki koşullar dikkate alınarak çalışmaktadır:

- C1: Düğüm nizami bir kök düğümdür (*ebeveyn = kendisi, kök = kendisi, uzaklık = 0*).
- C2: Düğüm ağaç içerisinde kök olmayan nizami bir düğümdür (*kök > kendisi, ebeveyn = komşusu, ebeveynin kökü = kendi kökü, uzaklık = ebeveynin uzaklığı + 1*).
- C3: Düğümün kökü komşuları içerisinde kökü en büyük olan düğümdür.

Bu koşullardaki ifadelerden uzaklık düğümün kök düğüme olan uzaklığını temsil etmektedir.

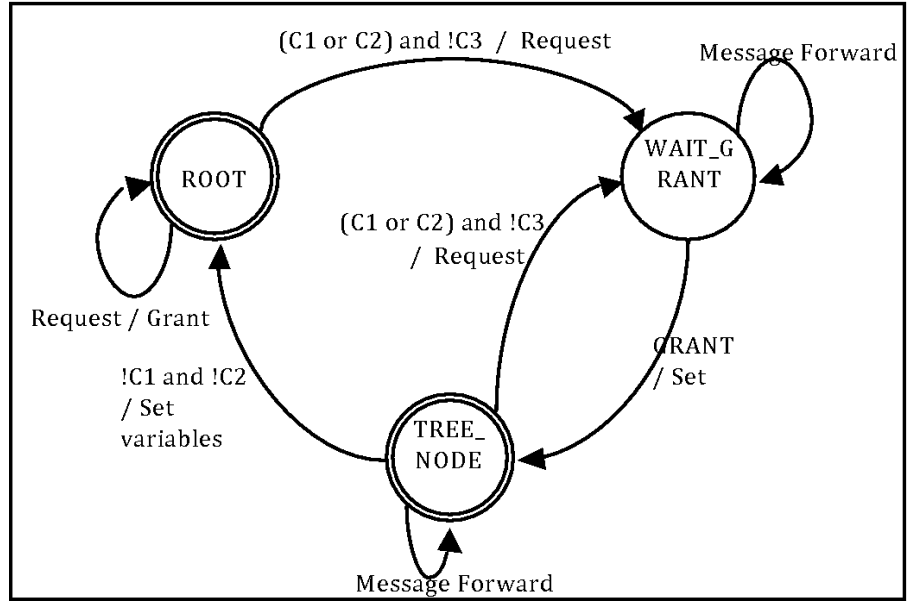
3.2.2.1 En büyük dereceli öz kararlı kapsayan ağaç algoritması 1 (EDKA1)

EDKA1 algoritmasında BVKA (Afek, et al., 1991) algoritması iyileştirilmiş ve kök düğümü seçimi yapılırken düğümlerin derecelerinin dikkate alınması sağlanmıştır. BVKA algoritması kapsayan ağacı oluştururken kök düğümü seçiminde düğümlerin kimlik numaralarını dikkate almaktaydı. Bu seçim kök düğüm olarak seçilecek düğümün çizgedeki özelliklerinin dikkate alınmaması sebebiyle karmaşık yapıdaki ağlarda rastgele sonuçlara sebep olabilmekteydi. Örneğin en büyük kimlik numarasına sahip düğümün bir yaprak düğüm olması durumunda bu düğüme olan tek bağlantı kolaylıkla bir darboğaz haline gelebilmekteydi. Bu gibi birçok durumda kök düğümün seçiminde düğümün çizgedeki özelliklerinin göz önünde bulundurulması gerekmektedir. Örneğin tüme gönderim işlemlerinin sıkça kullanılacağı sistemlerde kök düğüm seçilirken çizgenin orta noktası veya en yüksek dereceye sahip düğümün kullanılması yapılacak işlemlerin etkinliğini arttırabilmektedir. Bu sebeplerden dolayı BVKA algoritmasını en yüksek dereceli düğümü kök olarak seçecek şekilde geliştirilmesi önerilmektedir. Buradaki varsayım, birçok durumda kök olarak seçilmek üzere en yüksek dereceli düğümün gelişigüzel seçilmiş bir düğüme göre daha iyi bir seçim olacağı fikridir. Bu şekilde oluşturulacak kapsayan ağacın çapının azaltılması hedeflenmektedir. Bunun gerçekleştirilebilmesi amacıyla öncelikle düğümlerin

dereceleri ile kimlik numaralarının birleştirilip eşsiz bir etiket değeri oluşturan bir fonksiyon geliştirilmiştir. Bu fonksiyonda önceden belirlenmiş basamak sayısına sahip olan etiket değerinin en etkin kısmına düğümün derecesi, en az etkin kısmına da düğümün kimlik numarası yerleştirilmiştir. Burada ağdaki düğüm sayısının bir üst limiti geçmeyeceği varsayımı kullanılmış, etiketteki kimlik numarasının başına üst limitteki basamak sayısı kadar sıfır yerleştirilmiştir. Bu şekilde düğümlerin derecelerine göre sıralanmış eşsiz etiket değerleri elde edilmektedir. EDKA1 algoritmasında temel yöntem BVKA algoritmasıyla benzerlikler göstermektedir. BVKA algoritmasından farklı olarak kapsayan ağaç oluşturulurken düğümler kimlik numaraları yerine etiket numaralarını kullanmaktadır. EDKA1 algoritmasının sonunda en yüksek dereceli düğüm kök olacak şekilde bir kapsayan ağaç oluşmaktadır. Fakat yapılan bu değişiklik algoritmanın özkararlı yapısında da bazı değişiklikler yapılmasını gerektirmiştir. BVKA algoritmasında kapsayan ağaç oluşturulurken kullanılan kimlik değerlerinin değişmediği varsayılmaktadır. EDKA1 algoritmasında ise düğümlerin komşu sayılarının değişimiyle bağlantılı olarak dereceleri, dolayısıyla etiket değerleri de değişebilmektedir. Geçerli konumdaki bir ağda kök düğümün derecesinin değişmesi ağdaki diğer düğümlerin bunu fark etmemesine hatta derecesi değişen kök düğümün yeni kök olarak eski etiket değerindeki var olmayan bir düğümü seçmesine sebep olacaktır. Buna engel olabilmek amacıyla düğümlerin kök olarak seçtikleri düğümü değiştirecekleri zaman etiket değerinin kimlik kısmını kontrol etmeleri sağlanmıştır. Bu şekilde, yenilenen kök değerinin kimliğinin aynı olması durumunda düğümlerin kök etiketini güncellemeleri sağlanarak bu problemin önüne geçilmiştir.

EDKA1 algoritmasının sonlu konum diyagramı şekil 3.1’de görülebilmektedir. Her düğüm algoritmaya kendisini kök olarak seçerek başlamaktadır. Bunu takiben her düğüm algoritmadaki koşulları kontrol etmekte ve doğruluklarına göre işlem yürütmektedir. Eğer C1 veya C2 koşullarından biri doğru ve C3 hatalı ise bu düğümün daha büyük etiketli köke sahip bir komşusu olduğunu ifade etmektedir. Bu durumda düğüm en büyük köke sahip komşusunu belirleyip o komşu üzerinden kök düğüme *Request* tipinde bir mesaj göndermekte ve konumunu *WAIT_GRANT* olarak değiştirmektedir. Bu konumdaki bir düğümün *Grant* mesajı alması durumunda düğüm *TREE_NODE* konumuna geçmekte ve algoritmayı tamamlamaktadır. *TREE_NODE* konumundaki bir düğüm C1 ve C2 koşullarının yanlış olduğunu fark ederse kendisini kök olarak işaretlemekte ve konumunu *ROOT* haline getirmektedir. Bu algoritmanın

sonucunda sadece en büyük dereceli düğüm *ROOT* konumunda kalmakta, diğer bütün düğümler *TREE_NODE* konumuna geçmektedir.



Şekil 3.1 EDKA1 Algoritmasının sonlu konum diyagramı

3.2.2.2 En büyük dereceli öz kararlı kapsayan ağaç algoritması 2 (EDKA2)

EDKA1 algoritmasında kök seçimi ve düğümlerin kök değiştirmeleri seçilen kök düğüme gönderilen *request* ve *grant* mesajları kullanılarak yapılmaktaydı. EDKA2 algoritmasında mesaj karmaşıklığının azaltılması ve algoritmanın daha etkin çalışması hedeflenmektedir. Bu amaçla, EDKA1 algoritmasında kullanılan mesajlar kaldırılmakta ve düğümlerin kök seçimini mesaj gönderimi olmaksızın yapmaları sağlanmaktadır. EDKA1 algoritmasında kullanılan üç koşul (C1, C2 ve C3) EDKA2 algoritmasında da kullanılmaktadır. Bunlara ek olarak algoritmanın etkinliğinin artırılması için dördüncü bir koşul daha tanımlanmıştır. EDKA1 algoritmasında düğümlerin uzaklık değerlerinde olan bir değişiklik o düğümün altında bulunan tüm düğümlerin kendilerini kök ilan edip algoritmaya yeniden başlamalarına sebep olmaktaydı. Hâlbuki kök ve ebeveyn ilişkilerinin doğru olduğu fakat uzaklık değişkeninin hatalı olduğu durumlarda düğümlerin sadece uzaklık değerlerini düzeltmeleri sistemin geçerli konuma geçmesine yeterli olmaktadır. EDKA2 algoritmasında, bu durumun irdelenebilmesi için dördüncü bir koşul tanımlanmaktadır:

•C4: D ğ m k k olmayan nizami bir d ğ md r fakat uzaklık deęeri hatalıdır
 (k k > kendisi, ebeveyn = komşusu, ebeveynin k k  = kendi k k , uzaklık   ebeveynin uzaklıęı + 1).

EDKA2 algoritmasında her d ğ m kendisini k k se erek algoritmaya bařlamaktadır. Daha sonra C1, C2, C3 ve C4 kořulları kontrol edilmektedir. Eęer d ğ m n k k  komřuları arasında en b y k k k deęil ise (*C1 veya C2 doęru ve C3 yanlış*) k k n  en b y k k k olarak deęiřtirir, ebeveyni olarak ta bu k ke sahip olan komřusunu se er. Eęer d ğ mde C1 ve C2 kořullarının her ikisi de yanlışsa C4 kořulunu kontrol eder. Eęer C4 doęru ise d ğ m uzaklık deęerini ebeveyninkinden bir fazla olarak d zenler. Eęer C4 kořulu da yanlışsa d ğ m deęiřkenlerini sıfırlar ve kendisini k k olarak se er. Algoritma algoritma 3.1’de detaylarıyla g r lebilmektedir.

Algoritma 3.1. EDKA2 Algoritması

Girdi: *Komřu listesi*

Çıktı: *Kapsayan aęa *

```

1: eęer (C1 VEYA C2) VE !C3
2:     En b y k k ke sahip komřuyu bul
3:     eęer en b y k k k   kendi k k 
4:         ebeveyn   en b y k k ke sahip komřu
5:         k k   en b y k k k
6:         uzaklık   ebeveynin uzaklıęı + 1
7:         tip   D Đ M
8:         son
9: aksi halde !C1 VE !C2
10: eęer C4
11:     uzaklık   ebeveynin uzaklıęı + 1
12: aksi halde
13:     k k   kendi
14:     ebeveyn   kendi
15:     uzaklık   0
16:     tip   K K
17: son
  
```

18: son

19: son

3.2.2.3 En büyük dereceli merkez tabanlı öz kararlı kapsayan ağaç algoritması (EDMKA)

EDMKA algoritmasında, EDKA2 algoritmasında elde edilen kapsayan ağacın çapının azaltılması amacıyla ilerletilmesi hedeflenmiştir. Bunun için algoritmanın ağacın merkez düğümünü kök seçecek şekilde değiştirilmesi önerilmektedir. Buradaki varsayım, merkez düğümün kök olarak seçilmesinin ağacın çapını küçülteceği ve tüme gönderim gibi işlemlerde kolaylık sağlayacağıdır. Bunun gerçekleştirilebilmesi için EDKA2 algoritmasına öz kararlı bir merkez bulma algoritması eklenmektedir. Öz kararlılıkta geçerli olan varsayımların korunabilmesi için merkez bulma algoritması ile EDKA2 algoritmalarının sıralı olarak çalıştırılması önerilmektedir. Her bir turda önce EDKA2 algoritması ile kapsayan ağaç oluşturulmakta daha sonra merkez bulma algoritması ile ağacın merkezi bulunup düğüm etiketleri bu değer göz önüne alınarak güncellenmesi sağlanmıştır. EDMKA algoritmasında düğümlerin etiketleri üç değerın birleşimiyle elde edilmektedir, bunlar sırası ile yükseklik, derece ve kimlik değerleri olarak belirlenmektedir. Burada yükseklik düğümün en uzak düğüme olan uzaklığını, derece düğümün komşu sayısını kimlik düğümün mevcut kimlik değerini göstermektedir. EDMKA algoritmasında merkez bulma algoritması olarak Karaata ve arkadaşlarının öz kararlı merkez bulma algoritması kullanılmaktadır (Karaata, et al., 1994). Buna göre yaprak düğümler yükseklik değerlerini 0 olarak belirlerken diğer düğümler komşulukları içerisinde en büyük ikinci yüksekliğe sahip düğümün yüksekliğinden bir fazla olarak belirlemektedirler. Algoritmaya ait detaylar (Karaata, et al., 1994) çalışmasında gösterilmektedir.

EDMKA algoritması önce EDKA2 algoritmasını çalıştırmaktadır. Başlangıçta tüm düğümlerin yükseklik değerleri 0 olarak belirlenmiştir, bu sebeple algoritma ilk turda kapsayan ağaçtaki en büyük dereceli düğümü kök olarak seçecek şekilde çalışmaktadır. EDKA2 algoritması tamamlandıktan sonra ortaya çıkan kapsayan ağaç üzerinde öz kararlı merkez bulma algoritması çalıştırılıp ağacın merkez düğümü belirlenmektedir. Merkez düğüm yükseklik değeri en yüksek olan düğüm olarak işaretlenir. Burada ağaçtaki düğümlerin en büyük yükseklik değerini bilmelerine gerek yoktur, bunun yerine düğümlerin

etiket değerleri yükseklik değerleri de irdelenerek güncellenmektedir. Bu noktadan sonra EDKA2 algoritması en büyük etiket değerine, dolayısıyla en büyük yüksekliğe sahip düğümü kök olarak seçecek şekilde kapsayan ağacı günceller. Algoritmanın sonunda ilk oluşturulan kapsayan ağacın merkezi kök olacak şekilde bir kapsayan ağaç oluşturulmaktadır.

Bu algoritmanın katkısı merkez düğümün kök olarak seçilmesi ve bu şekilde oluşturulan kapsayan ağacın çapının düşürülmesidir. Bu şekilde kaynak taramasında kullanılacak mesajların ilinge üzerinde dağıtımının daha etkin biçimde gerçekleştirilmesi amaçlanmaktadır.

3.2.3 Analizler

Bu kısımda EDKA1, EDKA2 ve EDMKA algoritmaları için doğruluk ve karmaşıklık analizleri sunulmaktadır.

Kuram *EDKA1 algoritması sonlu sayıda adımda kökü en yüksek dereceli düğümlerden biri olan bir kapsayan ağaç oluşturmaktadır.*

İspat. EDKA1 Algoritması, (Afek, et al., 1991) çalışmasında ispatlanan BVKA algoritmasından esinlenerek oluşturulmuştur. BVKA algoritması, ağdaki her bir düğümün benzersiz kimlik numarası olduğunu varsaymaktadır. EDKA1 algoritmasında düğümlerin kimlik bilgisi yerine kimlik bilgisi ve derecesinin birleştirilmesinden oluşan etiket bilgisi kullanılmaktadır. Kullanılan etiket bilgisi eşsiz olacak şekilde oluşturulduğundan EDKA algoritması (Afek, et al., 1991) çalışmasında kullanılan ispat ile doğrulanmaktadır.

Kuram *EDKA2 algoritması sonlu sayıda adımda kökü en yüksek dereceli düğümlerden biri olan bir kapsayan ağaç oluşturmaktadır.*

İspat. EDKA2 algoritması EDKA1 algoritmasındaki mesajların kaldırılmış hali olarak tasarlanmıştır. Düğümlerin konum bilgileri dönemsel güncelleme mesajları kullanımı ile gerçekleştirilmektedir, dolayısı ile düğümlerden birinin konumu ile ilgili değişiklik sonlu sayıda tur sonunda düğümler arasında yayılmaktadır. Bu sebeple algoritma sonlu sayıda adımda kararlı hale geçmektedir.

Kuram EDMKA algoritması üçüncü turdan itibaren sonlu sayıda adımda kökü en yüksek dereceli olan ve ağacın merkezinde bulunan bir kapsayan ağaç oluşturmaktadır.

İspat. EDMKA algoritmasında etiket değerleri düğümün kimlik bilgisi, derecesi ve yükseklik değerini içermektedir. Yeni oluşturulan etiket değerleri eşsiz olduklarından daha önceki algoritmalar ile bir varsayım çakışması bulunmamaktadır. Algoritmanın ilk turunda en yüksek dereceli düğüm kök olacak şekilde bir kapsayan ağaç oluşmaktadır. İkinci turda ağacın merkezi bulunmakta ve tüm düğümlerin yükseklik değerleri güncellenmektedir. Üçüncü turda güncel etiket değerlerinin kullanılması sayesinde kök düğüm ağacın merkezi olacak şekilde kapsayan ağaç güncellenmektedir. Bu sebeple EDMKA algoritması üçüncü turdan itibaren EDKA2 algoritmasıyla aynı düzende çalışmaktadır. Bu noktadan itibaren algoritma sonlu sayıda adımda kökü merkez düğüm olan bir kapsayan ağaç oluşturmaktadır.

Kuram EDKA1 ve EDKA2 algoritmaları h kapsayan ağacın yüksekliği olacak şekilde h adımda kararlı hale gelmektedir.

İspat. EDKA1 ve EDKA2 algoritmalarında başlangıçta her düğüm kendisini kök düğüm olarak tanımlamaktadır. Takip eden ilk adımda en yüksek dereceli düğümün 1 zıplama mesafesindeki komşuları kararlı hale gelmekte ve değişkenlerini ayarlamaktadırlar. İkinci adımda kök düğümünden iki zıplama mesafesindeki düğümler kararlı hale gelmektedir. Oluşturulan kapsayan ağacın BFS tipinde bir kapsayan ağaç olması sebebiyle en son yaprak düğümler kararlı hale geleceklerdir. En uzaktaki yaprak düğüm kök düğümünden h zıplama mesafesinde bulunduğundan algoritma h adımda kararlı hale gelmektedir.

Kuram EDMKA algoritması h oluşacak kapsayan ağacın yüksekliği olacak şekilde $3h$ adımda kararlı hale gelmektedir.

İspat. EDMKA algoritmasının ilk turunda EDKA2 algoritması çalışmaktadır. İkinci turda merkez bulma algoritması çalışmakta ve ağacın merkezini belirlemektedir. Üçüncü turda ise EDKA2 algoritması tekrar çalışmakta ve sistem kararlı hale gelmektedir. EDKA2 ve merkez bulma algoritmalarının her ikisi de h adımda kararlı hale gelmesi sebebiyle EDMKA algoritması $3h$ adımda kararlı hale gelmektedir.

Kuram EDKA1 algoritmasının mesaj karmaşıklığı N düğüm sayısını ve h kapsayan ağacın yüksekliğini gösterecek şekilde $\Theta(2*N*h)$ olarak hesaplanmaktadır.

İspat. EDKA1 algoritmasının her adımında her bir düğüm değişkenlerini belirleyip iki mesaj değişimi ile komşularına iletmektedir. Sistemde N adet düğüm bulunması sebebiyle ve algoritmanın h adımda kararlı hale gelmesi sebebiyle algoritmanın mesaj karmaşıklığı $\Theta(2*N*h)$ şeklinde hesaplanabilmektedir.

Kuram EDKA2 algoritmasının mesaj karmaşıklığı N sistemdeki düğüm sayısını h kapsayan ağacın yüksekliğini gösterecek şekilde $\Theta(N*h)$ olarak gösterilmektedir.

İspat. EDKA2 algoritması h adımda kararlı hale gelmektedir. Her bir adımda her bir düğüm 1 adet dönemli mesaj gönderimine ihtiyaç duymaktadır. Sistemde N adet düğüm bulunması sebebiyle algoritma $N*h$ mesaj değişimi sonunda kararlı hale gelmektedir.

Kuram EDMKA algoritmasının mesaj karmaşıklığı, h kapsayan ağacın yüksekliğini, N sistemdeki düğüm sayısını gösterecek şekilde $\Theta(N*3h)$ olarak hesaplanmaktadır.

İspat. EDMKA algoritması $3h$ adımda kararlı hale gelmektedir. Her bir adımda düğüm başına 1 dönemli mesaj değişimi gerçekleşmektedir. Sistemde N düğüm olması sebebiyle toplamda $N*3h$ mesaj değişimi sonunda EDMKA algoritması kararlı hale gelmektedir.

3.2.4 Benzetimler

Bu kısımda, ikisi literatürde var olan (ZKKA ve BVKA), üçü yeni önerilen algoritmalar (EDKA1, EDKA2 ve EDMKA) olmak üzere toplam beş algoritma uygulanıp birbirleriyle karşılaştırılmaları sunulmaktadır. Uygulamalar ns2 benzetim ortamında geliştirilmiştir. Öz kararlı algoritmaların hata denetim sistemi periyodik mesajlarla benzetilmiştir. Bu amaçla her düğümün belli aralıklarla komşularına kendi değişkenlerini göndermesi sağlanmıştır. Bir düğüm bir komşusundan 3 fasıla boyunca bu mesajlardan almaz ise o komşunun ağdan ayrıldığı varsayılmıştır. Tüm algoritmalarda iletişimde UDP protokolü kullanılmıştır. Algoritmalarda, ns2'nun kablolu senaryolarda tüme gönderim

tipinde mesajları desteklememesi sebebiyle tüme gönderim tipinde mesaj kullanılmamıştır.

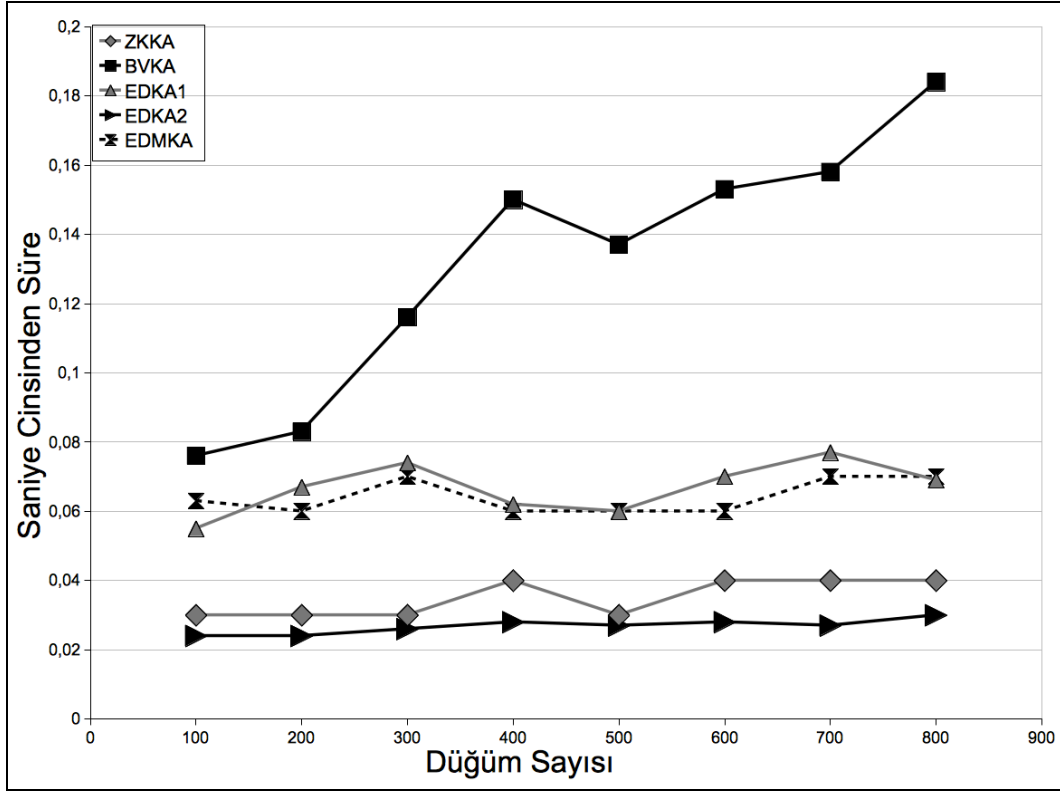
Deneylerde ns2 benzetim ortamı üzerinde 100'den 800'e kadar düğüm sayısına sahip 8 kablolu ağ senaryosu oluşturulmuştur. Öz kararlı algoritmalarda konum güncelleme mesajlarının sıklığının çalışma zamanını çok etkilediği fark edilmiştir, bu sebeple konum güncelleme mesaj gönderim aralığı ns2'nin izin verebileceği kadar düşük tutulmuştur. Algoritmalar arasında adil bir kıyaslama yapılabilmesi için bu güncelleme mesajlarının sıklığı tüm algoritmalarda eşit olarak kullanılmıştır.

Algoritmalara ait çalışma zamanı test sonuçları şekil 3.2'de görülebilmektedir. Şekilde de görülebileceği gibi en yüksek çalışma süresi BVKA algoritmasına aittir. Yeni geliştirilen algoritmalar BVKA'ye göre çalışma zamanı açısından daha iyi sonuçlar elde etmiştir. BVKA algoritmasında düğüm sayısına bağlı olarak çalışma zamanı artmaktadır. Bunun sebebi kök olarak seçilen düğümün ağ üzerinde gelişigüzel seçilmiş bir düğüm olmasıdır. Bu seçim kimi durumda düşük dereceye sahip bir düğümün kök olarak seçilmesine sebep olmakta, bu da hem darboğaz problemini ortaya çıkartmakta hem de oluşan kapsayan ağacın çapının büyük olmasına sebep olmasıdır. Bu problemler çalışma zamanında artışa yol açmaktadır. ZKKA algoritması öz kararlılık özelliğine sahip olmaması sebebiyle beş algoritma arasında en hızlı çalışan ikinci algoritmadır. Bunun sebebi öz kararlı algoritmalarındaki tersine fasılalı olarak çalışmamasıdır. Öz kararlı algoritmalar her fasılda sonuca yaklaşırken öz kararlı olmayan algoritmalar bir seferde işlemi tamamlamaktadır. Buna rağmen EDKA2 algoritmasında mesajlaşma kullanılmaması bu algoritmanın daha hızlı çalışmasına olanak vermiştir.

EDKA1 algoritması BVKA'ye göre daha iyi sonuçlar vermiştir, bunun sebebinin yüksek dereceli düğümün kök olarak seçilmesinin hem kapsayan ağacın çapını hem de darboğaz durumunu azaltması olduğunu düşünmekteyiz.

EDKA2 algoritması kapsayan ağaç oluşumunda mesaj iletişiminin kullanılmaması ve eklenen dördüncü koşul sebebiyle beş algoritma arasında en iyi sonuçları vermiştir. EDMKA algoritması ise EDKA2 algoritmasındaki temel fikir kullanılmasına rağmen EDKA2'den daha yavaş çalışmaktadır. Bunun sebebi algoritmanın merkez bulma algoritması ile fasılalı olarak çalışması ve kapsayan

ağacı kurduktan sonra kökün değişerek ağacın öz kararlı bir şekilde yeniden şekillendirilmesidir.

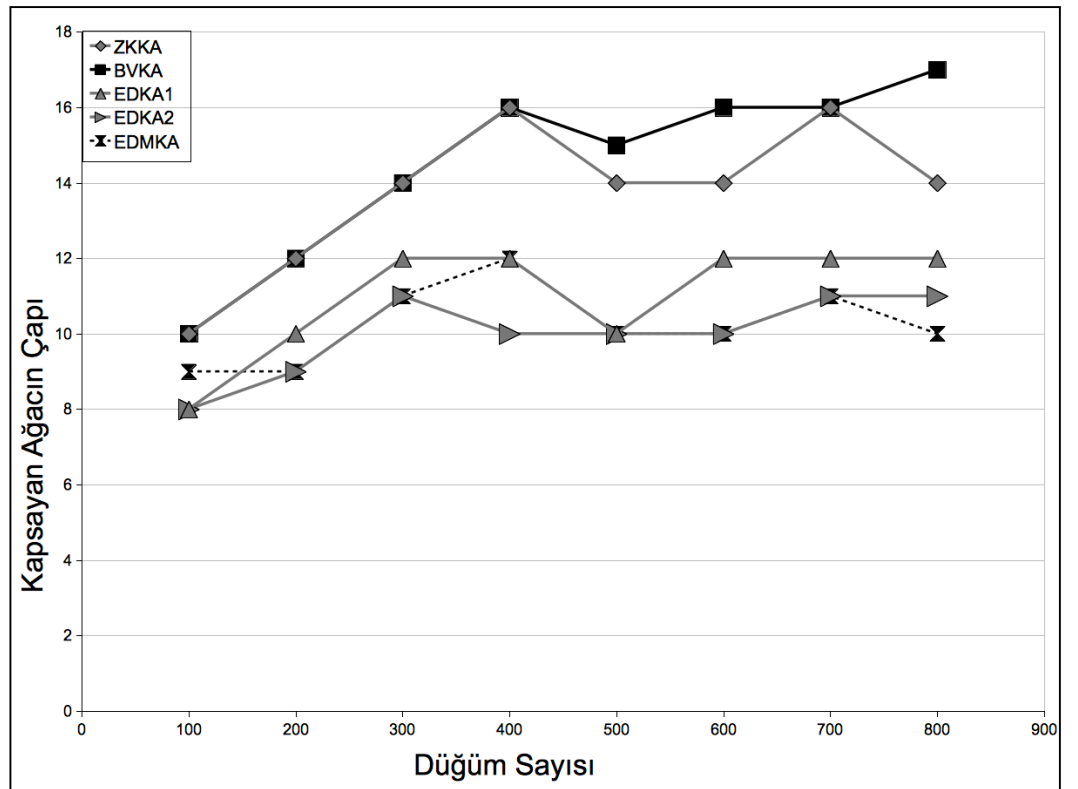


Şekil 3.2 Kapsayan ağaç oluşturma algoritmaları çalışma süreleri

Geliştirilen algoritmalar ile uygulanan deneylerde çalışma zamanının yanı sıra oluşturulan kapsayan ağacın çapı da incelenmiştir. Kapsayan ağacın çapının, oluşan kapsayan ağacı kullanacak olan uygulamalar için önemli olduğu düşünülmektedir. Bunun sebebi çapı küçük olan bir kapsayan ağaçta çoğa gönderim ve tüme gönderim işlemlerinin daha etkin olarak gerçekleşmesidir. Şekil 3.3'te görülebileceği gibi BVKA ve ZKKA algoritmaları benzer sonuçlar çıkartırken yeni geliştirilen algoritmalar daha küçük çaplı kapsayan ağaçlar elde etmişlerdir. Bunun sebebinin yeni geliştirilen algoritmalarda kullanılan dereceye bağlı kök seçimi olduğu düşünülmektedir. Yeni geliştirilen algoritmalar arasındaki farklar incelendiğinde, EDKA2 algoritmasının EDKA1 algoritmasına kıyasla daha iyi sonuçlar elde ettiği görülmektedir. Bunun sebebinin EDKA2 algoritmasında kullanılan dördüncü koşul olduğunu düşünülmektedir. Test sonuçlarında EDKA1 ve EDKA2 algoritmalarının her ikisinin de aynı kök düğümünü seçtikleri gözlemlenmiştir. Buna rağmen uzaklık değişkenlerindeki tutarsızlıkların ebeveyn ilişkisini bozmadan onarılması, kapsayan ağacın daha küçük çaplı olarak elde edilmesine sebep olmuştur. EDMKA algoritmasında diğer

algoritmalarla göre daha küçük çaplı kapsayan ağaçlar elde edilmesi beklenmekteydi. Fakat test sonuçları incelendiğinde EDMKA'nın beklenen iyileştirmeyi getirmediği görülmektedir. Bunun sebebi araştırıldığında çoğu durumda EDKA1 ve EDKA2 algoritmalarında oluşturulan kapsayan ağacın kökünün aynı zamanda o ağacın merkez düğümü olduğunu görülmüştür. Kullanılan senaryodaki ağ yapısının doğrusal ilinge tipinde bir yapıya benzerlik gösterdiği durumlarda EDMKA algoritması daha iyi sonuçlar vermektedir. Fakat karmaşık ilingelerde EDMKA'nın kullanımının kapsayan ağacın çapının büyüklüğü açısından gözle görülür bir iyileştirme sağlamadığını düşünülmektedir.

Şekil 3.2 ve şekil 3.3'teki sonuçlar göz önüne alındığında, yeni geliştirilen algoritmaların benzer nitelikteki var olan algoritmalarla kıyasla daha iyi sonuçlar elde ettiği görülmektedir. Yeni geliştirilen algoritmalar arasında da EDKA2 algoritmasının en iyi sonucu verdiği görülmektedir. Bu sonuçlar, başlangıçta yola çıkılan, en yüksek dereceli düğümün kök olarak kullanımının kapsayan ağacı daha etkin kılacağı varsayımını destekler nitelikte olduğunu göstermiştir. Buna karşılık merkez düğümün kök olarak seçiminin, karmaşık ağ ilingesinde başlangıçta düşünülen etkiyi yaratmadığı da test sonuçlarından gözlemlenebilmektedir.



Şekil 3.3 Kapsayan ağaç oluşturma algoritmalarında elde edilen kapsayan ağaçların çapları

3.2.5 Sonuç

Bu kısımda düğümlerin dereceleri göz önüne alınacak şekilde üç adet yeni öz kararlı kapsayan ağaç oluşturma algoritması önerilmiştir. Bunların yanı sıra önerilen algoritmalara benzer özelliklerde iki adet var olan kapsayan ağaç oluşturma algoritması ns2 ortamında uygulanıp, önerilen algoritmalarla çalışma zamanları ve oluşan kapsayan ağaçların çapları karşılaştırılmıştır. Bu beş algoritma arasındaki benzerlikler ve farklılıklar açıklanmış, önerilen algoritmaların getirdiği katkılar gösterilmiştir. Başlangıçta tahmin edilen derece değişkeninin kullanımından kaynaklanan iyileştirme test sonuçlarında gözlemlenmiş, öne sürülen fikir ile test sonuçları arasındaki tutarlılıklar ve tutarsızlıklar sebepleriyle açıklanmıştır.

Yapılan çalışmalar ve test sonuçları göz önüne alındığında MDST2 algoritmasının birçok durumda diğerlerinden üstün olduğu ve tezin geri kalanında MDST2 algoritmasının kullanımının uygun olduğu sonucuna varılmıştır.

3.3 Örgü Sistemlerde Sorgu İşleme İçin Kapsayan Ağaç Tabanlı Kaynak Tarama

Örgü sistemlerinde kaynak taraması sorguların etkin biçimde çalıştırılabilmesinde önemli adımlardan biridir. Örgü sistemlerinin özelliklerinden kaynaklanan problemler sebebiyle yeni kaynak tarama yöntemleri araştırılmaktadır. Örgü sistemlerde kaynak taraması bir görev için uygun olan düğümlerin örgü sistemlerinin özellikleri gözetilerek kısa sürede taranması olarak tanımlanmaktadır. Örgü sistemlerinin devingenlik ve geniş ölçeklilik özelliklerinden kaynaklanabilecek sorunların büyük bölümü kısım 3.2’de önerilen ilinge denetimi sayesinde çözülmektedir. Ancak güncel kaynak bilgilerinin güvenilir biçimde elde edilebilmesi amacıyla önerilen ilinge üzerinde bir kaynak tarama algoritması ihtiyacı bulunmaktadır.

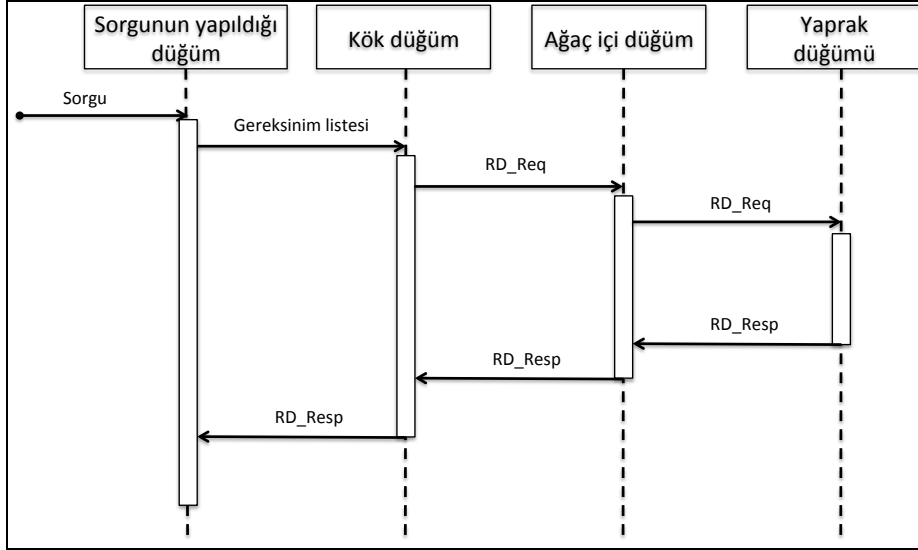
Örgü sistemlerde sorgu işleme için birçok kaynak tarama algoritması bulunmaktadır. Ancak örgü sistemlerin tüm özelliklerini ve sorguların gereksinimlerinin tamamını karşılayacak şekilde tasarlanmış kaynak tarama algoritmalarına bilimiz dâhilinde rastlanamamıştır. Daha kesin olmak gerekirse, örgü sistemdeki düğümlerin tümünün kaynak taraması işlemine katıldığı, ölçeklenebilir, güvenilir ve devingenliğe dayanıklı bir algoritma mevcut literatürde bulunmamaktadır. Bu sebeplerle bu kısımda EDKA2 algoritması

tarafından oluşturulan kapsayan ağacı kullanan bir kapsayan ağaç tabanlı kaynak tarama (KAKT) algoritması önerilmektedir.

3.3.1 Kapsayan ağaç tabanlı kaynak tarama (KAKT) algoritması

KAKT algoritmasında kapsayan ağaç şeklindeki ilingeden faydalanarak etkin bir biçimde kaynak taraması gerçekleştirilmesi hedeflenmektedir. Güncel ve güvenilir kaynak bilgisinin elde edilebilmesi amacıyla KAKT algoritması örgü sistemindeki tüm düğümlerle iletişim kurmaktadır. KAKT algoritmasının mevcut çalışmalar üzerine katkısı, kaynak tarama sorgularının tüme gönderilmesi ve sonuçların etkin biçimde geri toplanması olarak özetlenmektedir. KAKT algoritması tarafından kullanılan ilingede düğümler yalnızca kendi çocukları ve ebeveynleri ile iletişim kurmaktadır.

KAKT algoritması örgü sisteminde herhangi bir düğüme sorgu gönderilmesi ile başlamaktadır. Sorgunun yapıldığı düğüm sorguyu analiz etmekte ve kaynak özellik gereksinimlerini gereksinim listesi (GL) adı altında listelemektedir. Sorgunun yapıldığı düğüm GL'yi ebeveyni üzerinden kapsayan ağacın kök düğüme gönderir. Kök düğümü bu türde bir mesaj aldığında GL'yi *RD_Req* tipinde bir mesaj ile tüme gönderim yapılmak üzere çocuklarına gönderir. Mesaj yaprak düğümlere ulaşana kadar kapsayan ağaç üzerinde aşağılara doğru yayılmaktadır. Yaprak düğümler bu mesajı aldıklarında GL'de gösterilen gereksinimleri karşılayıp karşılamadıklarını kontrol etmektedir. Gereksinimlerin karşılanması halinde düğüm kendi kimlik bilgisini keşfedilen düğüm listesine (KDL) eklemektedir. Daha sonra KDL, *RD_Resp* tipinde bir mesaj içerisinde ebeveyne gönderilmektedir. Kapsayan ağaç üzerindeki ara düğümler tüm çocuklarından KDL'lerin toplanmasını beklemektedir. Tüm çocuklardan listeler alındıktan sonra düğüm gereksinimlerin karşılanması durumu için kendini kontrol eder ve kendi KDL'sini oluşturur. Daha sonra tüm KDL'ler tek bir KDL olacak şekilde birleştirilmekte ve ebeveyne gönderilmektedir. KDL'lerin birleştirilmesi sırasında keşfedilmiş düğümler arasındaki zıplama mesafesi de hesaplanıp KDL'nin içerisinde saklanmaktadır. KDL Kök düğüme eriştiğinde kaynak taraması tamamlanmış haldedir. Bu durumda kök düğüm KDL'yi sorgunun yapıldığı düğüme gönderir. KAKT algoritmasının akış diyagramı şekil 3.4'te gösterilmiştir.



Şekil 3.4 KAKT Algoritmasının akış diyagramı

3.3.2 Analizler

KAKT algoritması kapsayan ağaç yapısından faydalanmaktadır. Bu sebeple kapsayan ağaç yapısı ile birçok ortak özellikler taşımaktadır. Bu kısımda KAKT algoritması için güvenilirlik ve karmaşıklık analizleri sunulmaktadır.

Kuram KAKT algoritması 2. tür hata barındırmayan güncel kaynak bilgileri bulmaktadır.

İspat. KAKT algoritması öz kararlı bir kapsayan ağaç yapısı üzerinde çalışmaktadır. Kapsayan ağaç, tanım gereği sistemdeki tüm düğümleri içermektedir. Bu düğümlerin tümü KAKT algoritmasını çalışmasına katılmaktadır. KAKT algoritması süresince ilingenin sabit kaldığı varsayıldığında 2. tür hatalar söz konusu olmamaktadır. Ayrıca düğümlerin anlık kaynak bilgilerinin sunulması sebebiyle kaynak tarama sonuçları güncel verileri içermektedir.

Kuram KAKT algoritmasının mesaj karmaşıklığı N düğüm sayısını h kapsayan ağacın yüksekliğini gösterecek şekilde $\Theta(2h+2N)$ olarak gösterilmektedir.

İspat. KAKT algoritmasında sorgu öncelikle kök düğüme gönderilmektedir. Bu işlem h adet mesajın gönderilmesi ile gerçekleşmektedir. Daha sonra, kök düğüm bu mesajı yapraklara doğru tüme gönderim ile gönderir. Bu işlem N adet

mesaj gerektirmektedir. Yanıt mesajları yukarıya doğru tünden toplama yöntemi ile toplanmaktadır. Bu işlem de N mesaj gerektirmektedir. Son olarak kaynak tarama sonucunda elde edilen liste sorgunun yapıldığı düğüme gönderilir. Bu adım da h adet mesaj gerektirmektedir. Toplam olarak KAKT algoritması $2h+2N$ mesaj gerektirmektedir.

Kuram KAKT algoritmasının zaman karmaşıklığı h kapsayan ağacın yüksekliğini gösterecek şekilde $\Theta(4h)$ olarak hesaplanmaktadır.

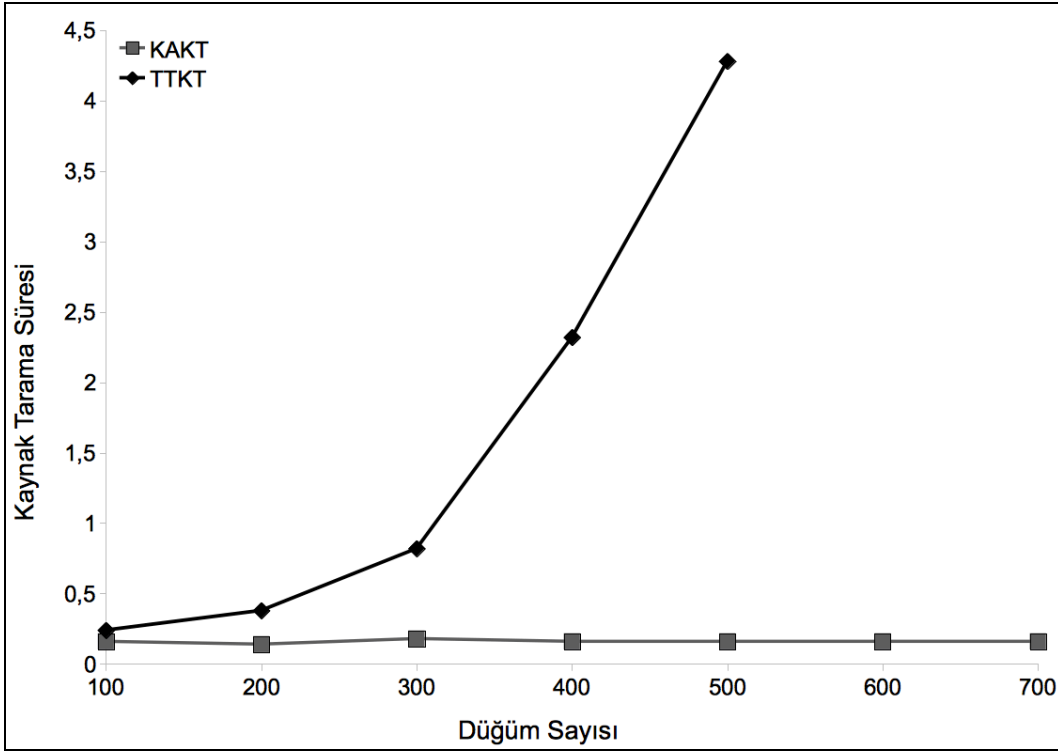
İspat. Kapsayan ağaç üzerindeki tüme gönderim ve tünden toplama işlemlerinin her biri koşut çalışmaları sebebiyle h adımda gerçekleşmektedir. Sorgunun kök düğüme gönderilmesi ve sonucun kök düğümden sorgulanan düğüme gönderilmesi işlemlerinin her biri de h adımda gerçekleşmektedir. Bu sebeple KAKT algoritması $4h$ adımda tamamlanmaktadır.

3.3.3 Benzetimler

Bu kısımda KAKT algoritması ns2 benzetim ortamında uygulanmakta ve deneyler yapılmaktadır. KAKT algoritmasıyla birlikte, karşılaştırma amacıyla literatürdeki benzer çalışmaların özelliklerini taşıyan taşıma tabanlı bir kaynak tarama (TTKT) algoritması da uygulanmaktadır. TTKT algoritmasında kaynak taramasında kullanılacak sorgular (Iamnitchi and Foster, 2003) çalışmasına benzer olarak örgü sistemine taşıma yöntemi kullanılarak dağıtılmaktadır. Kaynak gereksinimlerini karşılayan düğümler sorgunun yapıldığı düğüme cevap mesajı göndermektedir. KAKT ve TTKT algoritmaları kaynak taraması işleminin tamamlanma süreleri göz önüne alınarak karşılaştırılmaktadır.

Bu kısımda değerlendirilecek KAKT ve TTKT algoritmaları ns2 benzetim ortamında (Fall and Varadhan, 2010) uygulanmakta ve deneyler yapılmaktadır. Bu amaçla 100 ila 700 düğüm arasında örgü büyüklüğüne sahip, rastgele ağlardan oluşan 7 deney senaryosu hazırlanmıştır. KAKT algoritmasında kullanılan kapsayan ağaç önceden oluşturulmuştur. Her bir deney senaryosu için veri kaynağı olarak 20 düğüm rastgele olarak seçilmektedir. Deneylerde kaynak gereksinimi olarak sorguya katılan veri parçalarının bulunması koşulu kullanılmaktadır. Bununla birlikte kaynaklar keşfedilirken veri parçalarına sahip kaynakların özellikleri ve birbirlerine olan uzaklıkları da keşfedilmektedir.

Şekil 3.5’te, kaynak tarama işleminin tamamlanma süreleri ölçülmüştür. Sorgu, örgü sistemi üzerinde rastgele seçilmiş bir düğüme gönderilmiştir. Deney sonuçlarında görüldüğü üzere KAKT algoritmasının çalışma süresi test senaryosu boyunca neredeyse sabit kalmıştır. Buna karşılık TTKT algoritmasının çalışma süresi düğüm sayısı arttıkça yükselmektedir. Bunun sebebi KAKT algoritmasının zaman karmaşıklığının kapsayan ağacın yüksekliği ile bağlantılı olması ve TTKT algoritmasının zaman karmaşıklığının sistemdeki düğüm sayısı ile bağlantılı olmasıdır.



Şekil 3.5 Kaynak tarama algoritmaları tamamlanma süreleri

3.3.4 Sonuç

Bu kısımda, kapsayan ağaç ilingeleri üzerinde çalışan kapsayan ağaç tabanlı kaynak tarama (KAKT) algoritması önerilmiştir. Algoritma detaylı olarak açıklanmış, güvenilirlik ve karmaşıklık analizleri sunulmuştur. Bu analizler benzetim ortamında gerçekleştirilen deneylerle güçlendirilmiş ve benzer algoritmalarla karşılaştırmalar sunulmuştur.

Bu kısımda sunulan analizler ve deney sonuçları dikkate alındığında KAKT algoritmasının kısım 3.1’de belirtilen gereksinimlerin hemen tümünü karşıladığı söylenebilmektedir. Önerilen algoritma önceki kısımlarda önerilen öz-kararlı

kapsayan ağaç oluşturma algoritmaları ile ilişkili olarak çalışmaktadır. İlinge denetiminin öz kararlı olması sebebiyle KAKT algoritması örgü sistemlerinin devingenlik özelliğinden etkilenmemektedir. Kaynak taramasında kullanılan mesajların dağıtılmasında etkin tüme gönderim ve tümden toplama yöntemlerinin kullanılması ise ölçeklenebilirliği arttırdığı gibi tip 2 hataların oluşmasına da engel olmaktadır. Kaynak taramasında kullanılan mesajların tüm düğümlere ulaşması sayesinde aralıklı öznitelik, çoklu öznitelik ve devingen öznitelik taramalarının tümü yapılabilmektedir.

3.4 Genel Sonuçlar

Kaynak taraması örgü sistemlerde sorgu işleme için önemli bir adımı oluşturmaktadır. Özellikle örgü sistemleri söz konusu olduğunda geleneksel kaynak tarama yöntemlerinin üzerine sağlanması gereken birçok gereksinim bulunmaktadır. Bu gereksinimler bölüm 2’de detaylı olarak anlatılmıştır. Bölüm 2’de sunulan literatür taraması dikkate alındığında güncel ve güvenilir kaynak taramasının gerçekleştirilebilmesi için en etkin yöntemin tüm kaynakların kaynak taramasına katılması olduğu sonucuna varılmıştır. Ancak bu yöntem ölçeklenebilirlik problemleri ile karşı karşıya kalmaktadır. Literatürde önerilmiş bu yöntemi kullanan çalışmaların büyük bölümünde ölçeklenebilirlik problemine çözüm getirmek amacıyla kaynak tarama mesajlarının etki alanı kısıtlanmaktadır. Ancak bu kısıtlama tip 2 hataların oluşmasına ve buna bağlı olarak kaynak tarama yönteminin güvenilirliğinin azalmasına sebep olmaktadır. Ölçeklenebilirlik problemine sıradüzensel yapılanma ile çözüm getirmeyi amaçlayan çalışmalarda ise keşfedilen kaynak özelliklerinin güncel olmama problemi ortaya çıkmaktadır. Bunlar gibi literatürde birçok kaynak tarama yöntemi önerilmiştir. Her bir yöntemin kendine has fayda kazanımları ve fayda yitimleri bulunmaktadır. Örgü sisteminin özellikleri ve sorguların gereksinimlerini dikkate almak amacıyla ilingesel yapıları kullanan bilgimiz dâhilinde pek az çalışma bulunmaktadır. Bu sebeple bu bölümde önce öz kararlı kapsayan ağaç oluşturma algoritmaları daha sonra da kapsaya ağaç tabanlı kaynak tarama algoritması önerilmiştir. Önerilen algoritmalar için doğruluk ve karmaşıklık analizleri sunulmuştur. Her bir algoritma ve karşılaştırmalarda kullanılacak literatürde bulunan benzer algoritmalar ns2 benzetim ortamında uygulanmış ve önerilen algoritmalarla karşılaştırmaları verilmiştir. Önerilen algoritmaların güçlü ve zayıf yönleri tartışılmış, kullanılmalarının uygun olduğu durumlar açıklanmıştır.

Önerilen öz kararlı kapsayan ağaç oluşturma algoritmaları ile birlikte kapsayan ağaç tabanlı kaynak tarama algoritmasının, örgü sistemlerinin özelliklerini ve sorguların gereksinimlerini karşılayan bir algoritma olduğu gösterilmiştir. Önerilen yöntem devingenlik, kaynakların türdeş olmaması ve geniş ölçeklilik sorunlarının tümüne katkı sağlamaktadır. Ayrıca sorguların gereksinimlerinin tümüne çözüm önerilmiştir.

4. ÖRGÜ SİSTEMLERDE SORGU İŞLEME İÇİN KAYNAK ATAMASI

Bu bölümde örgü sistemlerde sorgu işleme için düğümlerin veri kaynaklarına olan uzaklıklarını dikkate alan bir kaynak atama algoritması önerilmektedir. Bunun için öncelikle aday düğüm listesi arama uzayı, adayların veri kaynaklarına olan uzaklıkları gözetilerek sınırlandırılmaktadır. Daha sonra aday düğümler arasından sorgu işleme sırasındaki veri iletişimini azaltacak şekilde kaynak ataması gerçekleştirilmektedir. Algoritmalar detaylı anlatımlarla önerilmekte, analizler sunulmakta ve benzetim ortamında gerçekleştirilen deneylerle değerlendirmeleri ve benzer çalışmalar ile karşılaştırmaları sunulmaktadır.

4.1 Giriş

Örgü sistemleri geniş ölçekli hesaplama kabiliyetleri ve yüksek erişilebilirlikleri sebebiyle günümüzün ilgi çeken araştırma konularından biri haline gelmiştir (Foster and Kesselman, 2004). Bu sebeple birçok güncel araştırma alanı örgü sistemlerinin sunduğu olanaklardan faydalanmaktadır. Dağıtık sorgu işleme, mevcut altyapısını koşut ve dağıtık sistemlerden örgü sistemlere taşımak amacıyla yüksek miktarda araştırmalar yapılan alanlardan birisidir (Huang, et al., 2003; Antonioletti, et al., 2005; Pacitti, et al., 2007; Kotowski, et al., 2008; Taniar, et al., 2008). Ancak, örgü sistemlerin özellikleri bu aşamada birçok sorunun ortaya çıkmasına sebep olmaktadır. Örgü sistemleri, koşut ve dağıtık sistemlerden farklı olarak geniş ölçekli, türdeş olmayan ve devingen olarak nitelendirilmektedir (Foster and Kesselman, 2004). Örgü sistemlerde genellikle yüksek miktarlarda kaynak olduğu varsayılmaktadır. Bu kaynaklar hesaplama kaynakları, veri kaynakları veya hizmet kaynakları gibi farklı kavramlara karşılık gelebilmektedir. Buna karşın, düğümler örgü sistemi içerisindeki bilgisayarlara karşılık gelmektedir. Örgü sistemlerindeki bu düğümler farklı özelliklerde ve devingen olabilmektedir. Düğümlerin devingenlikleri özelliklerinin devingen olmasının yanı sıra örgü sistemlerindeki varlıklarının devingenliğine de karşılık gelebilmektedir. Örgü sisteminde düğümler herhangi bir zamanda haber vermeden sisteme katılıp sistemden ayrılabilir (Foster and Kesselman, 2004). Dağıtık sorgu işleme tarafından bakıldığında, örgü sistemlerinin bu özellikleri kaynak tarama ve kaynak ataması gibi sorunların ortaya çıkmasına sebep olmaktadır. Örgü sisteminde dağıtık sorgunun verimli şekilde işlenebilmesi için öncelikle dağıtılmış haldeki verinin bulunduğu

düğümlemlerin ve dağıtık veriye ait ön bilginin bulunması gerekmektedir. Bu gibi bilgiler kaynak tarama adımıyla elde edilmektedir. Bunun için gerekli olan kaynak tarama algoritmaları bölüm 3'te sunulmuştur. Kaynak tarama aşaması gerçekleştirildikten sonra aday kaynakların uygun biçimde atanması sorgu işleminin başarımını doğrudan etkilemektedir. Kaynak ataması aday kaynaklardan kaç adedinin ve hangilerinin sorgu işleminde kullanılacağını belirlemektedir. Literatürde bu konuda gerçekleştirilmiş birçok çalışma bulunmaktadır (Gounaris, et al., 2004; Gounaris, et al., 2005; Soe, et al., 2005; Gounaris, et al., 2006; Silva, et al., 2006; Bose, et al., 2007; Kotowski, et al., 2008; Liu and Karimi, 2008; Gounaris, et al., 2009). Bu çalışmaların birçoğu literatür taraması çalışmaları içerisinde sınıflandırma ile de incelenmiştir (Costa and Furtado, 2008; Epimakhov, et al., 2011; Cokuslu, et al., 2012a). Her ne kadar mevcut kaynak atama algoritmaları bu soruna başarılı çözümler üretmiş olsalar da, aday düğüm örnek uzayının büyüklüğü hiçbir çalışmada dikkate alınmamıştır. Örgü sistemler gibi geniş ölçekli sistemlerde aday düğüm örnek uzayı aday düğümlerin tümünün değerlendirilmesine engel olabilecek boyutlarda olabilmektedir. Bu sebeple aday düğümler incelenirken örnek uzayın küçültülmesi gerekmektedir. Bilgimiz dâhilinde literatürde kaynak atama sırasında aday düğüm arama uzayının büyüklüğünü azaltan çalışmalar bulunmamaktadır. Bunun yanı sıra mevcut kaynak ataması çalışmaları içerisinde çok azı kaynak seçiminde iletişim maliyetleri üzerine odaklanmıştır (Bose, et al., 2007; Liu and Karimi, 2008). Özellikle sorgu işleminde düğümler arasında aktarılan verinin büyüklüğü göz önüne alındığında iletişim maliyetlerinin öncelikli olarak irdelenmesi gerekmektedir. Bütün bunlar göz önüne alındığında, örgü sistemler üzerinde sorgu işlemesi için kaynak ataması konusunda halen açık konuların bulunduğu düşünülmektedir.

Bu bölümde örgü sistemleri üzerinde sorgu işleme için kaynak atama algoritmaları önerilmesi amaçlanmıştır. Bunun için öncelikle tek bir birleştirme işletmeninden oluşan sorgulara kaynak ataması yapan tekli birleştirme işletmeni kaynak atama (TBİKA) algoritması (Cokuslu, et al., 2012b) önerilmektedir. TBİKA'da birleştirme işletmeni için sınırlı sayıda bir aday düğüm örnek uzayı oluşturulmaktadır. Sonrasında, aday düğümlerin veri kaynaklarına uzaklıkları göz önüne alınarak artımlı şekilde bir kaynak atama modeli oluşturulmaktadır. TBİKA algoritmasından sonra, önerilen algoritma geliştirilmekte ve çoklu birleştirme işletmeni kaynak atama (ÇBİKA) algoritması önerilmektedir. ÇBİKA algoritmasında sorgu ağacı sondan başlayarak gezilmekte ve görevlerin türüne

göre TBİKA algoritmasından da faydalanarak kaynak ataması gerçekleştirilmektedir.

Tezin bu bölümünde önerilen algoritmaların bilime katkısı iki yönlüdür. Birincisi, aday düğüm örnek uzayının büyüklüğünün kısıtlanması sayesinde kaynak atama algoritmasının ölçeklenebilirliği artırılmaktadır. İkinci katkı ise kaynak atama için düğüm seçiminde iletişim maliyetlerinin irdelenmesidir.

Bu bölümün devamında, birleştirme işletmenine girdi olan ilişkisel verilerin yatay olarak bölümlendiği ve her bir bölümün yalnız bir düğümde bulunduğu varsayılmaktadır. Mevcut bölümlenme, birleştirmede kullanılan alana göre yapılmış olmayabilir. Bu sebeple birleştirme süresince yeniden bölümlenme gerekebilmektedir. Bu çalışmada birleştirme işletmeninin iki temel görevden oluştuğu varsayılmaktadır, bunlar *tarama* ve *birleştirme* görevleri olarak adlandırılmaktadır. *Tarama* görevi veriyi depolama biriminden okuyup ilgili *birleştirme* görevine göndermekle yükümlüdür. Veri yerelliği göz önüne alındığında *tarama* görevlerinin, verilerin bulunduğu düğümlere atandığı varsayılmaktadır. Bu düğümler çalışmanın devamında *tarama düğümleri* olarak adlandırılacaktır. *Birleştirme* görevlerinin atanacağı düğümler *birleştirme düğümleri*, sorgunun yapıldığı düğüm ise *sorgu düğümü* olarak adlandırılacaktır.

Bu bölümün geri kalanı şu şekilde yapılanmıştır. Kısım 4.2’de Tekli Birleştirme İşletmeni Kaynak Atama Algoritması önerilmektedir. Kısım 4.3’te Çoklu Birleştirme İşletmeni Kaynak Atama Algoritması önerilmektedir. Son olarak kısım 4.4’te, sonuçlar tartışılmaktadır.

4.2 Tekli Birleştirme İşletmeni Kaynak Atama (TBİKA) Algoritması

Bu bölümde, tek birleştirme işletmeninden oluşan sorgular için *tarama* ve *birleştirme* görevlerine kaynak atama yapan TBİKA algoritması (Cokuslu, et al., 2012b) önerilmektedir. Veri yerelliği kısıdı göz önüne alındığında, *tarama* görevinin verinin bulunduğu kaynaklara atanması öngörülmektedir. Bu sebeple bu bölümde önerilen algoritma *birleştirme* görevi için uygun kaynak ataması gerçekleştirmektedir. *Birleştirme* görevinin atanabileceği aday düğümler için aranacak özellikler kısıtlıdır ve büyük bir aday düğüm örnek uzayı oluşmasına sebep olabilmektedir. Bu sebeple TBİKA algoritması birbiriyle tamamlayıcı olan iki alt algoritmadan oluşmaktadır. İlk önce sezgisel olarak küçültülmüş bir aday

düğüm uzayı oluşturulması hedeflenmektedir. Bu amaçla ilk olarak kısım 4.2.1’de Yakınlık Tabanlı Aday Düğüm Listesi Oluşturma (YADLO) algoritması önerilmektedir. Sonlu sayıdaki aday düğüm listesi oluşturulduktan sonra da kısım 4.2.2’de Birleştirme Görevi Kaynak Atama (BGKA) algoritması önerilmektedir.

4.2.1 Yakınlık tabanlı aday düğüm listesi oluşturma (YADLO) algoritması

Birleştirme görevi için kaynak ataması, örgü sistemindeki her düğümün aday olabilmesi sebebiyle karmaşık bir işlemdir. Geniş ölçekli bir sistemde tüm düğümleri değerlendirmeye katmak kaynak atamasını zaman alıcı bir işlem haline dönüştürmektedir. Bu sebeple *YADLO* algoritması ile aday düğüm sayısının kısıtlanmasını hedeflemekteyiz. Kaynak atama algoritmamızın temelinde fikir olarak iletişim maliyetlerinin azaltılmasının olması sebebiyle, *YADLO* algoritmasında aday düğüm sayısının küçültülebilmesi için adayların veri kaynaklarına olan uzaklıklarının kullanılmasının faydalı bir yöntem olduğunu düşünmekteyiz. Bu sebeple *YADLO* algoritması ile aday düğüm listesi oluşturulurken *tarama düğümlerine* yakın olan düğümlerin seçilmesi hedeflenmiştir. Bunun gerçekleştirilmesi için tüm *tarama düğümlerinden* bir taşıma mesajı gönderilmektedir. Taşıma mesajları örgü üzerinde yayılırken bu mesajları alan düğümler mesajı gönderen düğümü kaydetmektedirler. *Tarama düğümlerinin* tümünden mesaj alan düğüm, *sorgu düğümüne* kendisini aday olarak tanıtmaktadır. *Sorgu düğümü* üzerinde *YADLO* algoritması algoritma 4.1’de gösterilmektedir. Algoritma ilk olarak tüm *tarama düğümlerine* *YADLOBaşlat* mesajı göndererek taşımanın başlatılmasını sağlar (satır 1). Mesaj gönderildikten sonra *sorgu düğümü* aday düğümlerin cevaplarını bekler. Aday düğümlerden cevap mesajları geldikçe *sorgu düğümü* adayları listeye ekler (satır 3). Listeye ilk eklenen düğüm tüm *tarama düğümlerine* en yakında bulunan düğümdür. Satır 4’te, algoritma listeyi toplamaya devam edip etmeyeceğine karar vermektedir. *Sorgu düğümünün* taşıma işleminin ne zaman biteceğinden haberi olmaması sebebiyle algoritma sezgisel olarak sonlandırılmaktadır. *YADLO* algoritmasında sonlandırma kısıtı olarak *tarama düğümleri* alt kümesinin çapı (h) kullanılmaktadır. *Tarama düğümleri* alt kümesinin çapı, tezimizin bir önceki aşaması olan kaynak taraması tarafından verilmektedir. Burada kaynak tarama kapsayan ağacı üzerinde birbirine en uzak olan iki *tarama düğümü* (*düğümA* ve *düğümB*) arasındaki uzaklık kullanılmaktadır. Bu iki düğümden herhangi birinden *YADLOAday* tipinde bir mesaj alınması, taşırılan mesajın h adım ilerlemiş olduğunu göstermektedir. Bu da taşırılan mesajın en azından *tarama düğümlerinin*

tümüne ulaştığını göstermektedir. Bu gibi bir durumda *sorgu düğümü* algoritmayı sonlandırır.

Algoritma 4. 1. Sorgu düğümü üzerinde YADLO Algoritması

Girdi: (i) Tarama düğümleri listesi

(ii) Birbirine en uzak iki tarama düğümü (*düğümA*, *düğümB*)

Çıktı: Aday düğüm listesi

1: tüm tarama düğümlerine *YADLOBaşlat* mesajı gönder

2: *YADLOAday* mesajı alınması halinde:

3: gönderen düğümü aday listesine ekle

4: **eğer** *gönderen* = *düğümA* veya *düğümB* **ise**

5: *algoritmayı sonlandır*

6: **aksi halde**

7: *YADLOAday* mesajı almaya devam et

8: **son**

Sorgu düğümü dışındaki diğer düğümlerde çalıştırılacak *YADLO* algoritması algoritma 4.2’de gösterilmektedir. Örgü sistemdeki bir düğüm bir mesaj aldığı anda mesajın tipini kontrol etmektedir. Eğer mesaj tipi *YADLOBaşlat* ise düğüm komşularına *YADLOTaşıma* tipinde bir mesaj göndererek taşıma işlemini başlatır (satır 3). Algoritma için sonlandırma kısıtı bu mesaj içerisinde *zıplamaLimiti* değeri içerisinde tutulmaktadır. Bu değer *tarama düğümleri* altkütlesinin çapına eşittir ve kaynak tarama adımı tarafından sağlanmaktadır. Eğer düğüm tarafından alınan mesajın tipi *YADLOTaşıma* ise alıcı düğüm, mesajın kaynağından uzaklığını gösteren *zıplamaSayacı* değerini 1 artırır (satır 6). Saha sonra mesajın kaynağını *mesajAlınanTaramaDüğümleri* listesine ekler (satır 7). Eğer *mesajAlınanTaramaDüğümleri* listesi tüm *tarama düğümlerini* kapsıyorsa, bu düğüm *sorgu düğümüne* *YADLOAday* mesajı göndererek kendi adaylığını bildirir (satır 9). Daha sonra aynı düğüm mesajın taşınmaya devam edilip edilmeyeceğini kontrol eder (satır 11); eğer sonlandırma kısıtı henüz karşılanmamış ise mesaj komşulara iletilir (satır 12). Taşıma mesajları örgü üzerinde *tarama düğümlerinden* *zıplamaLimiti* değeri kadar uzağa dağıtılmaktadır. Sezgisel olarak belirlenmiş olan bu kural taşıma mesajlarının en azından tüm *tarama düğümlerine* ulaşmasını garanti etmektedir. Bu durumda en

kötü halde, t tarama düğümü sayısı olacak şekilde, t adet aday düğüm elde edilecektir.

Algoritma 4.2. Sorgu düğümü dışındaki düğümlerde YADLO Algoritması

- 1: Mesaj alınması durumunda:
 - 2: **eğer** *mesaj tipi YADLOBaşlat ise*
 - 3: *tüm komşulara YADLOTaşıma mesajı gönder*
 - 4: **aksi takdirde eğer** *mesaj tipi YADLOTaşıma ise*
 - 5: *mesaj içerisinden zıplamaLimiti ve zıplamaSayacı değerlerini oku*
 - 6: *zıplamaSayacı değerini bir artır*
 - 7: *mesajAlınanTaramaDüğümleri listesine mesajın kaynağını ekle*
 - 8: **eğer** *mesajAlınanTaramaDüğümleri listesi tüm tarama düğümlerini kapsıyor ise*
 - 9: *sorgu düğümüne YADLOAday mesajı gönder*
 - 10: **son**
 - 11: **eğer** *zıplamaSayacı < zıplamaLimiti ise*
 - 12: *mesajı gönderen dışındaki tüm komşulara YADLOTaşıma mesajını ilet*
 - 13: **aksi takdirde**
 - 14: *mesajı taşırmayı durdur*
 - 15: **son**
 - 16: **son**
 - 17: **son**
-

4.2.2 Birleştirme görevi kaynak atama (BGKA) algoritması

Sorguların bölümlenmiş koşutluk ile işletilmesinde en temel çözüm *birleştirme* görevinin tek bir düğüm üzerinde çalıştırılmasıdır. Öte yandan *birleştirme* görevinin birden çok düğüm üzerine yayılarak koşut çalıştırılması sorgu işlemenin başarımını önemli ölçüde etkilemektedir. Buna ek olarak örgü üzerindeki kaynakların türdeş olmaması, sorgu işlemede kullanılacak ideal düğüm

sayısının önceden belirlenmesini zorlaştırmaktadır. Bu sebeple *birleştirme* görevi için kaç adet düğüm atanacağını belirlenmesi önem taşımaktadır. Bu gibi ortamlarda sorgu işlemenin başarımını etkileyen çok miktarda değiştirge bulunmaktadır. Düğümlere ait işlemci hızı, bellek miktarı, ağ bağlantı hızı, yük durumu gibi düğüm özellikleri değiştirgelere örnek sayılabilir. Kaynak atama algoritmasında bu değiştirgelerin tümünün irdelenmesi çok maliyetli olabilmektedir. Bu sebeple mevcut çalışmaların genelinde kullanılacak değiştirgeler sezgisel olarak seçilmektedir (Garofalakis and Ioannidis, 1997; Gomoluch and Schroeder, 2003; Slimani, et al., 2004; Mandal, et al., 2005; Bose, et al., 2007; Liu and Karimi, 2008). Örgü üzerinde sorgu işlenirken *tarama* ve *birleştirme düğümleri* arasında yoğun veri aktarımı olmaktadır. Bu aktarım örgü üzerinde görece yavaş iletişim kanalları üzerinden gerçekleşmektedir. Bu sebeple biz örgü sistemler üzerinde sorgu işlemenin başarımını belirleyen etmenin veri iletişimi maliyetleri olduğunu düşünmekteyiz. Buradan yola çıkarak, çalışmamızın bu bölümünde veri iletişim maliyetlerini göz önüne alan bir kaynak atama algoritması önermekteyiz. Önerdiğimiz algoritma (Gounaris, et al., 2006) çalışmasındaki algoritmayla benzer şekilde artımlı olarak kaynak ataması yapmakta ve başarımlarının durduğu noktada algoritmayı sonlandırmaktadır. (Gounaris, et al., 2006) çalışmasından farklı olarak *BGKA* algoritmasında kaynak ataması *tarama düğümlerinin* yakınındaki aday düğümlerden başlanarak gerçekleştirilmektedir. Burada iletişim maliyetlerinin azaltılması hedeflenmektedir. *BGKA* algoritmasındaki bir diğer katkı da atanacak düğüm sayısının belirlenmesinde kullanılan bir karar fonksiyonunun önerilmesidir. Karar fonksiyonu düğümler arasında veri iletişim sürelerini tahminleyerek koşutluk derecesinin artırılıp artırılmayacağına karar vermektedir.

BGKA algoritmasında *YADLO* algoritması tarafından oluşturulan aday listesindeki aday düğümler kullanılmaktadır. Bu listenin başında *tarama düğümlerinin* tümüne en yakın olan aday düğüm bulunmaktadır. Listede aşağılara doğru inildikçe *tarama düğümlerinden* uzaklaşmakta ve iletişim kanallarının hızları azalmaktadır. *BGKA* algoritması algoritma 4.3'te gösterilmektedir. Algoritma bir döngü içerisinde aday listesinden atanan kaynaklara yeni düğümler eklemektedir (satır 2). Satır 3'te, listeden alınan düğümün *tarama düğümlerine* olan iletişim hızı hesaplanır. Bu hesaplama ağ tarafından sağlanan *RTT* gibi servisler aracılığı ile gerçekleşir. Yeni düğümün eklenmesi, *birleştirme düğümlerinin* her birine aktarılacak olan verinin azalmasına sebep olmaktadır. Ancak yeni düğümler eklendikçe eklenen düğümün daha uzak olması sebebiyle iletişim hızı azalmaktadır. Bu da birleştirme işletmeninin başarımındaki

hızlanmanın azalarak artmasına sebep olmaktadır. Yeni düğümlerin eklenmesi, hızlanma miktarı sıfırın altına düşene kadar devam etmektedir.

Algoritma 4.3. BGKA Algoritması

Girdi: (i) Birleştirme görevi aday düğüm listesi (*adayListesi*)

(ii) Sorguya katılan ilişkiler ile ilgili ön bilgi

Çıktı: Birleştirme görevi için atanmasına karar verilen düğüm listesi

1: **döngü**

2: *denemeDüğümü* $\leftarrow$ *adayListesi*'nden düğüm çıkart

3: *denemeDüğümü* ile tüm tarama düğümleri arasındaki iletişim hızlarını ölç

4: seçilen düğümler ile birleştirme işlemi sırasında gerçekleşecek veri aktarım süresini tahminle

5: **eğer** yeni tahmin < önceki tahmin **ise**

6: *denemeDüğümü*'nü *seçilenDüğümListesi*'ne ekle

7: **son**

8: *adayListesi* boşalana kadar döngüyü tekrarla

10: **son**

BGKA Algoritmasında veri aktarım süresinin tahminlenmesinde üç değişirge dikkate alınmaktadır: (i) tarama ve birleştirme düğümleri arasındaki iletişim hızı (S_{ij}), (ii) her bir birleştirme düğümünün yerel bağlantı hızı (B_j), (iii) tarama düğümlerinde bulunan ilişki bölümlerinin büyüklükleri ($|P_i|$). Bu değişirgelerden B_j ve $|P_i|$ kaynak taraması tarafından verilmektedir. Ancak S_{ij} *BGKA* algoritmasının çalışması sırasında yeni seçilen her bir aday düğüm için hesaplanmaktadır. Her aday düğüm için hesaplanan S_{ij} değeri anlık olarak hatasız olsa da, sorgu işlemesi sırasındaki başarıyı yansıtmayabilmektedir. Çünkü bu ölçüm yapıldığı sırada sorgu henüz atanmamıştır. Sorgu atandıktan sonra düğümler birden çok bağlantı kuracağı için gerçek *RTT* değerleri ölçülenden farklı olacaktır. Bu hatanın sezgisel olarak azaltılabilmesi için veri aktarım süresi hesaplanırken algoritma 4.4'te gösterilen normalizasyon yöntemini önermekteyiz. Buna göre, bir düğüm ile kurulan bağlantılar için ölçülen *RTT* değerlerinin toplamı, düğümün yerel bağlantı hızını aşması halinde, *TCP*'deki akış kontrol

mekanizmasına benzer şekilde, tahminlenen *RTT* değerleri oransal olarak azaltılmaktadır.

Algoritma 4.4. Bağlantı hızı tahmini normalizasyon algoritması

Girdi: (i) *Tarama ve birleştirme düğümleri arasında ölçülen RTT değerleri* (S_{ij})
(ii) *Seçilen birleştirme düğümünün yerel bağlantı hızı* (B_j)

Çıktı: S_{ij} 'lerin normalleştirilmiş listesi

```

1:  $toplam_j \leftarrow \sum_{i=0 \rightarrow n} S_{ij}$ 
2:  $fark \leftarrow B_j - toplam_j$ 
3: eğer  $fark > 0$  ise
4:   döngü  $i = 0$ 'dan  $n$ 'e kadar
5:      $S_{ij} = S_{ij} - (fark * (S_{ij} / toplam_j))$ 
6:   son
7: son
8: son

```

Tarama düğümleri üzerindeki ilişki bölümlerinin tekbiçimli dağılım (*ing. uniform distribution*) gösterdiği varsayıldığında n adet *tarama düğümü* ve m adet *birleştirme düğümü* arasındaki koşut veri iletişimi süresi en yavaş iletişim kanalı ile belirlenmektedir. Bu şekildeki bir sistemde her bir i *tarama düğümü* ile *birleştirme düğümleri* arasında iletilecek veri miktarı, $|P_i|$ *tarama düğümündeki* ilişki bölümünün büyüklüğünü gösterecek şekilde, $(|P_i| / m)$ olarak hesaplanmaktadır. Bu sebeple bu sistemdeki veri iletişim süresi denklem (1)'deki gibidir.

$$\max_{\substack{i=0 \rightarrow n \\ j=0 \rightarrow m}} \frac{|P_i|}{m * S_{ij}} \quad (1)$$

Denklem (1) en yavaş iletişim kanalını belirlemektedir. Bu şekilde birleştirme görevinin tahmini süresini hesaplamaktadır. *BGKA* algoritmasında yeni aday düğümlerin atanmış düğüm listesine eklenip eklenmeyeceği bu denkleme bakılarak kararlaştırılmaktadır.

4.2.3 Analizler

Bu bölümde, önerdiğimiz *YADLO* ve *BGKA* algoritmaları için karmaşıklık analizleri verilmektedir.

Kuram *YADLO* algoritmasının zaman karmaşıklığı, d ağın çapını gösterecek şekilde, $O(d)$ olarak hesaplanır.

İspat. *YADLO* algoritması sonlanma koşulu olarak birbirine en uzak iki tarama düğümü arasındaki mesafeyi kullanmaktadır. En kötü senaryo halinde bu iki düğüm birbirinden ağın çapı (d) kadar uzaklıktadır. Bu durumda algoritma d adım sonra sonlanmaktadır. Buna göre algoritmanın zaman karmaşıklığı $O(d)$ olarak hesaplanır.

Kuram *YADLO* algoritması, n tarama düğümü sayısını, N ise örgü sistemindeki toplam düğüm sayısını gösterecek şekilde, $O(nN^2)$ mesaj karmaşıklığına sahiptir.

İspat. *YADLO* algoritmasında, her bir tarama düğümü kendinden başlayan bir taşıma işlemi başlatmaktadır. Bu sebeple sistemde aynı anda n adet taşıma işlemi yürütülmektedir. Bir taşıma işleminin mesaj karmaşıklığı $O(N^2)$ olduğuna göre, *YADLO* algoritmasının mesaj karmaşıklığı $O(nN^2)$ olarak hesaplanabilir.

Kuram *BGKA* algoritmasının zaman karmaşıklığı, n sistemdeki tarama düğümü sayısını, N ise örgü sistemindeki toplam düğüm sayısını gösterecek şekilde, $O(nN)$ olarak hesaplanmaktadır.

İspat. *BGKA* algoritması, *YADLO* algoritması tarafından oluşturulan aday kaynak listesini kullanmaktadır. En kötü durumda, bu liste ağdaki tüm düğümleri kapsamaktadır. *BGKA* algoritmasında bu aday düğümlerin her biri için n adet tarama düğümüne olan uzaklıkları ölçülmektedir. Bu sebeple *BGKA* algoritmasının zaman karmaşıklığı, n tarama düğümü sayısını, N ise örgü sistemindeki düğüm sayısını ifade edecek şekilde, $O(nN)$ olarak hesaplanmaktadır.

Kuram *BGKA* algoritmasının mesaj karmaşıklığı, n tarama düğümlerinin sayısını, N ise örgü sistemindeki toplam düğüm sayısını gösterecek şekilde, $O(2nN)$ 'dir.

İspat. BGKA algoritması aday düğümler ile tarama düğümleri arasındaki bağlantı hızının ölçülmesi sırasında 2 adet mesaj kullanılmaktadır. N adet aday düğüm için n adet tarama düğümlerine olan uzaklıklar hesaplanırken $O(2nN)$ adet mesaj kullanılmaktadır.

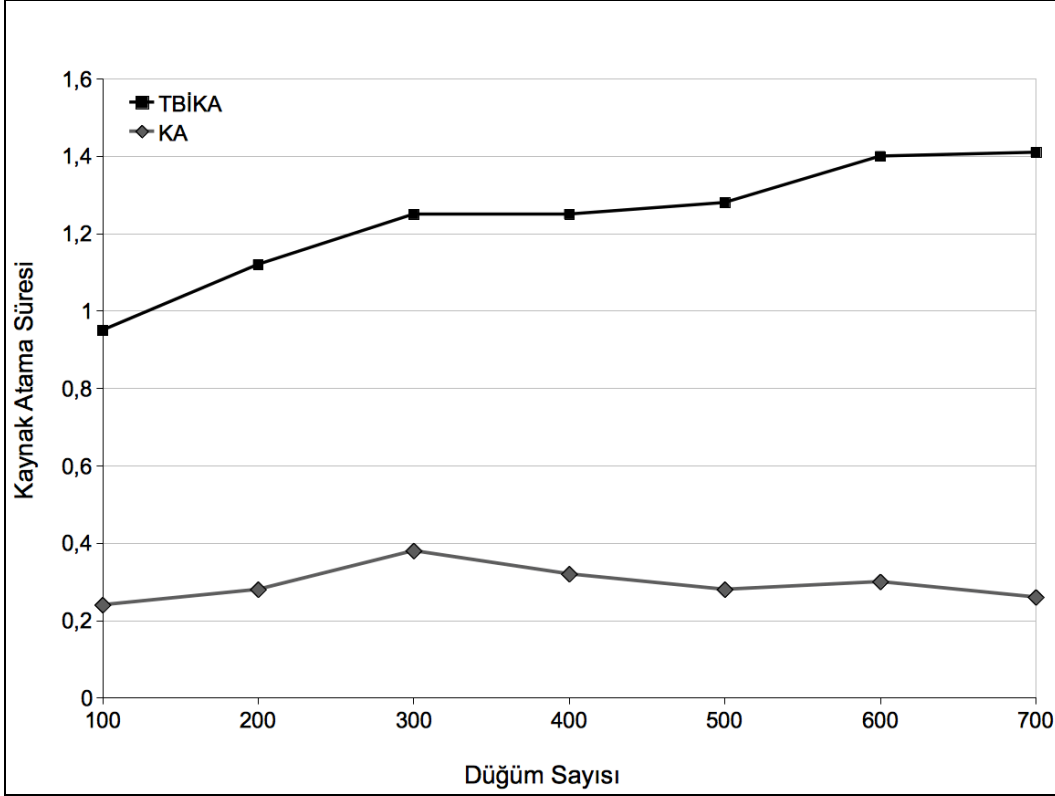
4.2.4 Benzetimler

Bu kısımda TBİKA algoritması için benzetim ortamında gerçekleştirilen deney sonuçları ile niceliksel değerlendirmeler sunulmaktadır. TBİKA algoritması, benzer özelliklerdeki mevcut kaynak atama algoritmalarının ortak özelliklerini taşıyan, (Gounaris, et al., 2004; Gounaris, et al., 2006) çalışmalarındaki benzer bir kıyaslama algoritması (KA) ile karşılaştırılmaktadır. KA'daki ana fikir Gounaris ve arkadaşları tarafından sunulmuş (Gounaris, et al., 2004; Gounaris, et al., 2006) çalışmalarındaki yöntemle aynıdır. Algoritma örgü sistemindeki düğümleri özelliklerine göre değerlendirip sıralamaktadır. Bu kısımda gerçekleştirilmiş olan deneylerde düğüm özellikleri olarak işlemci güçleri ve düğümlerin bağlantı hızları kullanılmaktadır. KA, sıralanmış düğümleri en yüksek puanlıdan başlayacak şekilde kaynak atamasını gerçekleştirmekte ve her bir düğüm eklendiğinde başarımlar tahminini hesaplamaktadır. Başarım tahminindeki artış durduğunda algoritma tamamlanmakta ve o ana kadar seçilmiş düğümlerin kaynak atamasını gerçekleştirmektedir. TBİKA ve KA algoritmaları ns2 benzetim ortamında uygulanmıştır. Kaynak atama işleminin ve atanan kaynaklar kullanılarak çalıştırılan örnek bir sorgunun süreleri ölçülmüş, her iki algoritmanın başarımları karşılaştırılmıştır.

Benzetimlerde 100 ila 700 arasında düğümden oluşan örgü sistemleri kullanılmaktadır. Senaryolardaki her bir düğüm tek işlemcili bir bilgisayarı temsil etmektedir. Her bir düğümün komşuları ve komşularına olan bağlantı hızları rastgele olacak şekilde belirlenmektedir. Bağlantı hızları 1 ve 10 Gbps arasında olacak şekilde atanmaktadır. Senaryolarda veri kaynakları örgü sistemi üzerinden rastgele seçilen 20 düğüm üzerinde olacak şekilde belirlenmiştir. Her bir veri kaynağı, üzerinde sorguya katılacak veritabanı ilişkisinin bir bölümünü barındırmaktadır. Bölümlerin büyüklükleri 50 GByte olacak şekilde belirlenmiştir.

Şekil 4.1'de TBİKA ve KA ile gerçekleştirilen kaynak atama işlemi için gerekli olan süreler gösterilmektedir. Şekilden görülebileceği üzere TBİKA algoritması kaynak ataması için daha uzun süreye ihtiyaç duymaktadır. Bunun

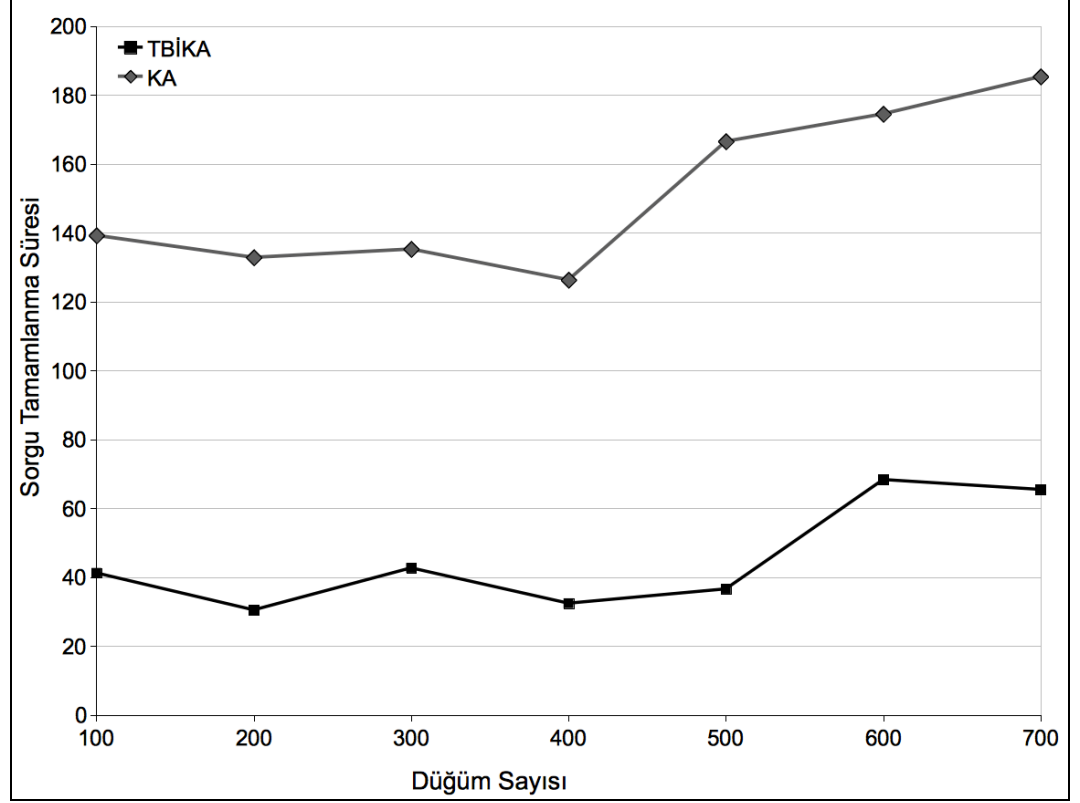
sebebi TBİKA algoritmasında iletişim maliyeti tahminleri hesaplanırken tüm aday düğümlerin veri kaynaklarına olan uzaklıkları ölçülmektedir. Buna karşılık KA algoritmasında bu ölçüm yapılmamaktadır. Ayrıca örgü sistemindeki düğümlerin tamamının irdelenmesinin engellenebilmesi amacıyla KA algoritmasında sorgu işleminin başarımının artışı durduğu anda algoritması durmaktadır. Bu sebeple KA algoritması daha kısa sürede tamamlanmaktadır. Buna karşılık bu yöntem ile daha iyi başarımla sağlama olasılığı olan düğümler irdelenmemektedir.



Şekil 4.1 Kaynak ataması için gerekli süreler

Şekil 4.2 BİKA ve KA algoritmaları kullanılarak atanmış kaynakların sorgu işleminde katılmaları halinde sorgu işleme sürelerini göstermektedir. Şekilde görülebileceği üzere TBİKA algoritması tarafından gerçekleştirilen kaynak ataması sonucunda işlenen sorgunun tamamlanması için geçen süre KA algoritması ile atanan kaynaklardan daha kısadır. Bunun sebebi, KA algoritması tarafından seçilen kaynakların veri kaynaklarına olan uzaklıklarının ve iletişim maliyetlerinin hesaba katılmamasıdır. Buna karşılık TBİKA algoritması veri kaynaklarına yakın düğümleri atamaktadır. Bu şekilde sorgu işlemesi sırasında iletişim maliyetleri düşürülmektedir. Dağıtık sorguların veri iletişimi açısından baskın olması sebebiyle örgü sistemleri üzerinde sorgu işleminde TBİKA algoritması KA algoritmasından daha iyi sonuçlar elde etmiştir.

Şekil 4.1 ve 4.2’de gösterilen deney sonuçları dikkate alındığında, TBİKA algoritmasının başlangıç kaynak ataması süresinin uzun olması karşılığında sorgu işlemede daha iyi sonuçlar vermektedir. Bu sebeple sorguların uzun sürdüğü, veri kaynaklarının dağınık bulunduğu geniş ölçekli sistemlerde TBİKA algoritmasının tercih edilebilir olduğu gözlemlenmektedir.



Şekil 4.2 Sorgu işleme süreleri

4.2.5 Sonuç

Bu kısımda tek bir birleştirme işletmenine aday düğümlerin veri kaynaklarına olan uzaklıklarını dikkate alarak kaynak ataması gerçekleştiren tekli birleştirme işletmeni kaynak atama (TBİKA) algoritması önerilmiştir. Algoritma detaylı olarak anlatılmış, karmaşıklık analizleri sunulmuştur. TBİKA algoritması bir karşılaştırma algoritması ile birlikte ns2 benzetim ortamında uygulanmış ve çeşitli karşılaştırmalar yapılmıştır. TBİKA algoritmasının literatürde mevcut algoritmaların özelliklerini taşıyan karşılaştırma algoritmasına karşı zayıf ve güçlü yönleri tartışılmış ve TBİKA algoritmasının kullanılmasının uygun olduğu durumlar anlatılmıştır.

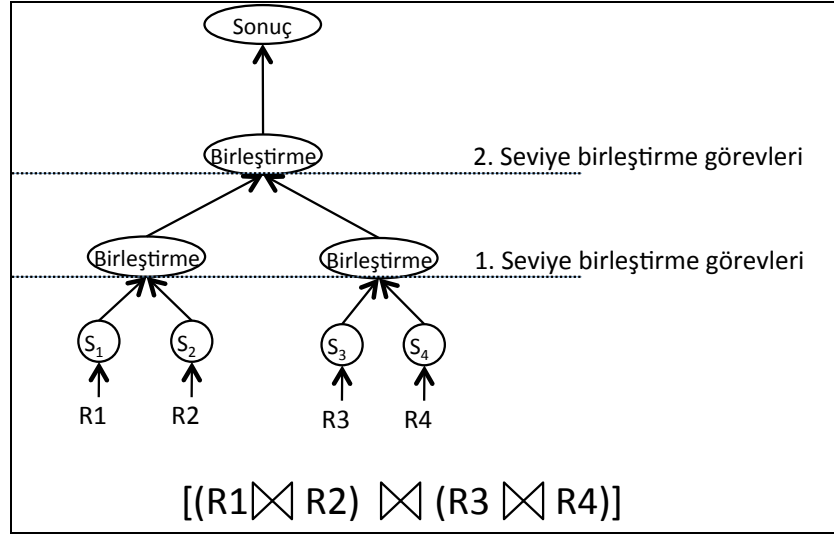
4.3 Çoklu Birleştirme İşletmeni Kaynak Atama (ÇBİKA) Algoritması

Bu kısımda TBİKA algoritmasının çoklu birleştirme işletmenlerinden oluşan sorgularda çalışacak şekilde genişletilmiş hali olan ÇBİKA algoritması önerilmektedir. ÇBİKA algoritmasında birden çok birleştirme işletmeninden oluşan sorguların bütününün kaynak atamasının gerçekleştirilmesi hedeflenmektedir. Çoklu birleştirme işletmeninden oluşan sorgularda oluşacak olan ara ilişkilerin hangi düğümler üzerinde bulunacağı önceden bilinmemektedir. Bu sebeple ÇBİKA algoritmasında sorgu ağacının sondan başlayarak gezilmesi ve görevlerin türüne göre kaynak atamasının yapılması önerilmektedir. Bu kısmın geri kalanı şu şekilde yapılanmaktadır: kısım 4.3.1’de ÇBİKA algoritması önerilmektedir. Kısım 4.3.2’de önerilen algoritmanın karmaşıklık analizleri sunulmaktadır. Kısım 4.3.3’te analizler benzetim ortamında gerçekleştirilen deney sonuçları ile doğrulanmaktadır. Son olarak kısım 4.3.4’te sonuçlar sunulmaktadır.

4.3.1 Çoklu birleştirme işletmeni kaynak atama (ÇBİKA) algoritması

Kısım 4.2’de önerilen TBİKA algoritmasında tek bir birleştirme işletmenine ait görevlere kaynak ataması gerçekleştirilmiştir. Tezimizde irdelenen sorguların bir veya birden çok birleştirme işletmeni içerdiği düşünüldüğünde TBİKA algoritmasının, sorgudaki tüm görevleri kapsayacak şekilde geliştirilmesi gerekmektedir. Bu bölümde, birden çok birleştirme işletmeninden oluşan sorgu içerisindeki görevlerin tümü için kaynak ataması gerçekleştiren çoklu birleştirme işletmeni kaynak atama (ÇBİKA) algoritması önerilmektedir. ÇBİKA algoritmasında sorguların ağaç gösterimleri kullanılmaktadır. Örnek bir ağaç gösterim şekil 4.3’te gösterilmektedir. Şekildeki ağaçtaki köşeler görevleri, aralarındaki kenarlar da görevler arasındaki veri akışını temsil etmektedir. ÇBİKA algoritmasında sorguların kodlayarak birleştirme işletmenlerinden oluştuğu varsayılmaktadır. Kodlayarak birleştirme işletmenlerinin çalışmasına dair detaylı bilgi (Özsu and Valduries, 2011) çalışmasında gösterilmektedir. Birleştirme işletmenleri *tarama*, *inşa* ve *yoklama* görevlerinden oluşmaktadır. Her bir birleştirme işletmeni, sorguya katılan ilişkilerin farklı bölümleri üzerinde koşut olarak çalışan birden çok *inşa* ve *yoklama* görevi ikilisinden oluşabilmektedir. Aynı ilişki bölümü üzerinde çalışan *inşa* ve *yoklama* görev ikilileri birbirleri ile yoğun etkileşim içerisinde çalışmaktadır. Bu sebeple aynı veri bölümü üzerinde çalışan *inşa* ve *yoklama* görev ikililerinin aynı düğüm üzerinde çalıştırılması

gerekmektedir. Çalışmamızın devamında, ilişkilerin aynı bölümleri üzerinde çalışan *inşa* ve *yoklama* görev ikilileri birleştirme görevi olarak adlandırılacaktır.



Şekil 4.3 Örnek bir sorgunun ağaç gösterimi

Tezimizin önceki bölümünde önerilen TBİKA algoritmasında ilişkilerin bölümlerinin bulunduğu düğümlerin kaynak tarama adımı sonucunda öğrenildiği kabul edilmektedir. Ancak geçici sonuçların olduğu bir sorgunun işletilmesi sırasında oluşan geçici ilişkilerin hangi düğümlerde bulunacağı baştan bilinmemektedir. Bu sebeple ÇBİKA algoritmasında kaynak ataması sorgu ağacının en altında bulunan birleştirme görevlerinden başlamak üzere gerçekleştirilecektir. Bu görevler Şekil 4.3'te 1. seviye birleştirme görevleri olarak gösterilmiştir. 1. seviye görevlerin kaynak ataması tamamlandıktan sonra kaynak ataması ağaç üzerinde yukarıya doğru devam edecektir. Bu amaçla sorgu ağacı sondan başlayarak gezilerek kaynak ataması gerçekleştirilecektir. ÇBİKA algoritması sorgunun gönderildiği düğüm tarafından çalıştırılmaktadır. Başlangıç kaynak ataması tamamlandığında görevlere atanan düğümlerin listesi sorgu ağacı içerisinde tutulmaktadır. Sorgu ağacı üzerindeki her bir köşe içerisinde *görevElemanı* adında bir veri yapısı bulunmaktadır. Bu veri yapısının içeriği Şekil 4.4'te gösterilmektedir. *görevElemanı* veri yapısı içerisinde *görevTipi*, *düğümListesi*, *sağİşaretçi* ve *solİşaretçi* olmak üzere 4 adet alan bulunmaktadır. *görevTipi* alanı sorgu ağacı üzerindeki köşenin görev tipini (*tarama* veya *birleştirme* görevi) göstermektedir. Algoritmanın başlangıcında içi boş olan *düğümListesi* alanı görev için atanmış olan düğümlerin listesini tutmaktadır. *sağİşaretçi* ve *solİşaretçi* alanları ise köşenin sağ ve sol çocuklarını işaret etmektedir.

görevElemanı
• <i>integer</i>: görevTipi • <i>array</i>: adayListesi • <i>pointer</i>: solİşaretçi • <i>pointer</i>: sağİşaretçi

Şekil 4.4. Sorgu ağacının köşelerini temsil eden veri yapısı

ÇBİKA algoritması algoritma 4.5'te gösterilmektedir. Algoritma özyinelemeli olarak sondan başlayarak ağaç gezinmesi gerçekleştirmektedir (satır 1 - 6). Satır 7'de algoritma ziyaret edilen köşeyi işleme almaktadır. Eğer ziyaret edilen köşenin görev tipi *tarama* ise, düğüm listesi alanı taramanın gerçekleştirileceği tablonun bölümlerinin bulunduğu düğümler ile doldurulur. Bu bilgi hâlihazırda kaynak tarama adımı tarafından verilmektedir (satır 8). Eğer ziyaret edilen köşenin görev tipi *birleştirme görevi* ise, düğüm listesi alanı TBİKA algoritmasına gerekli değiştirgelerin verilmesi sonucu elde edilen liste ile doldurulur. Bu adımda gerekli olan tablo bölümlerinin bulunduğu düğüm listeleri çocuk köşeler tarafından sağlanmaktadır (satır 10). Sorgu sırasında oluşan ara tabloların büyüklüklerinin tahminlerinin dışarıdan sağlandığı varsayılmaktadır. Ara tabloların bölümlerinin bulunduğu düğümler arasında birbirine en uzak olan iki düğümün kimlikleri ve aralarındaki uzaklık bilgisi (*düğümA*, *düğümB* ve *uzaklık*) yine tezimizin kaynak tarama bölümünde önerilen ilinge denetimi aracı tarafından anlık olarak sağlanmaktadır. ÇBİKA algoritmasının sonucunda sorgu ağacındaki tüm köşeler ilgili görevler için seçilmiş olan düğümlerin listeleri ile dolu olmaktadır.

Algoritma 4.5. ÇBİKA Algoritması

Girdiler: *Sorgu ağacının kök düğümü*

Çıktılar: İşlenmiş haldeki sorgu ağacı

- 1: ÇBİKAAlgoritması(ağaçKöşesi)
 - 2: **Eğer** ağaçKöşesi boş **ise**
 - 3: *geri dön*
 - 4: **Aksi halde**
 - 5: ÇBİKAAlgoritması(ağaçKöşesi.solİşaretçi)
 - 6: ÇBİKAAlgoritması (ağaçKöşesi.sağİşaretçi)
 - 7: **Eğer** ağaçKöşesi.görevTipi = *tarama* **ise**
 - 8: ağaçKöşesi.düğümListesi = *tablo bölümlerinin bulunduğu düğümler*
 - 9: **Aksi halde** **eğer** ağaçKöşesi.görevTipi = *birleştirme görevi* **ise**
 - 10: ağaçKöşesi.düğümListesi =
-
- TBİKAAlgoritması(ağaçKöşesi.solİşaretçi.düğümListesi) ∪*
-

ağaçKöşesi.sollşaretçi.düğümListesi, düğümA, düğümB, uzaklık, tabloBüyüklikleri)

11: **son**

12: **son**

13: *baş'a dön*

14: **son**

4.3.2 Analiz

Bu kısımda ÇBİKA algoritmasının zaman ve mesaj karmaşıklığı analizleri verilmektedir.

Kuram ÇBİKA algoritmasının zaman karmaşıklığı j birleştirme işletmeni sayısını, n veri kaynağı sayısını, N de sistemdeki düğüm sayısını gösterecek şekilde $O(jnN)$ olarak hesaplanmaktadır.

İspat. ÇBİKA algoritması birleştirme işletmenlerinin atanması için TBİKA algoritmasını kullanmaktadır. TBİKA algoritmasının zaman karmaşıklığının $O(nN)$ olması sebebiyle, j adet birleştirme işletmeninden oluşan bir sorgu için ÇBİKA algoritmasının zaman karmaşıklığı $O(jnN)$ olarak hesaplanmaktadır.

Kuram ÇBİKA algoritmasının mesaj karmaşıklığı j birleştirme işletmeni sayısını, n veri kaynağı sayısını ve N sistemdeki düğüm sayısını temsil edecek şekilde $O(jnN^2)$ olarak gösterilmektedir.

İspat. ÇBİKA algoritması birleştirme işletmenlerinin atanması için TBİKA algoritmasını kullanmaktadır. TBİKA algoritmasının zaman karmaşıklığının $O(nN^2)$ olması sebebiyle, j adet birleştirme işletmeninden oluşan bir sorgu için ÇBİKA algoritmasının zaman karmaşıklığı $O(jnN)$ olarak hesaplanmaktadır.

4.3.3 Benzetimler

Bu kısımda ÖBİKA algoritması ile ilgili benzetim ortamında gerçekleştirilen deney sonuçlarına dayalı değerlendirmeler ve karşılaştırmalar sunulmaktadır. ÇBİKA algoritması, mevcut algoritmaların ortak özelliklerini taşıyan (Gounaris, et al., 2004; Gounaris, et al., 2006) çalışmasına benzer bir karşılaştırma algoritması (KA) ile karşılaştırılmaktadır. KA'daki ana fikir Gounaris ve arkadaşları tarafından sunulmuş (Gounaris, et al., 2004; Gounaris, et

al., 2006) çalışmalarındaki yöntemle aynıdır. Algoritma örgü sistemindeki düğümleri özelliklerine göre değerlendirip sıralamaktadır. Bu kısımda gerçekleştirilmiş olan deneylerde düğüm özellikleri olarak işlemci güçleri ve düğümlerin bağlantı hızları kullanılmaktadır. KA, sıralanmış düğümleri en yüksek puanlıdan başlayacak şekilde kaynak atamasını gerçekleştirmekte ve her bir düğüm eklendiğinde başarımlı tahminini hesaplamaktadır. Başarımlı tahminindeki artış durduğunda algoritma tamamlanmakta ve o ana kadar seçilmiş düğümlerin kaynak atamasını gerçekleştirmektedir. ÇBİKA ve KA algoritmaları ns2 benzetim ortamında uygulanmıştır. Kaynak atama işleminin ve atanan kaynaklar kullanılarak çalıştırılan örnek bir sorgunun süreleri ölçülmüş, her iki algoritmanın başarımları karşılaştırılmıştır.

Benzetimlerde 100 ila 700 arasında düğümden oluşan örgü sistemleri kullanılmaktadır. Senaryolardaki her bir düğüm tek işlemcili bir bilgisayarı temsil etmektedir. Her bir düğümün komşuları ve komşularına olan bağlantı hızları rastgele olacak şekilde belirlenmektedir. Bağlantı hızları 1 ve 10 Gbps arasında olacak şekilde atanmaktadır. Deney senaryolarında 3 adet birleştirme işletmeni barındıran 4 adet taban ilişkisi birleştiren bir örnek sorgu kullanılmaktadır. Örnek sorgunun biçimsel sunumu şekil 4.5'te gösterilmektedir. Burada R1, R2, R3 ve R4 taban ilişkileri, $\bowtie$ ise birleştirme işletmenini temsil etmektedir.

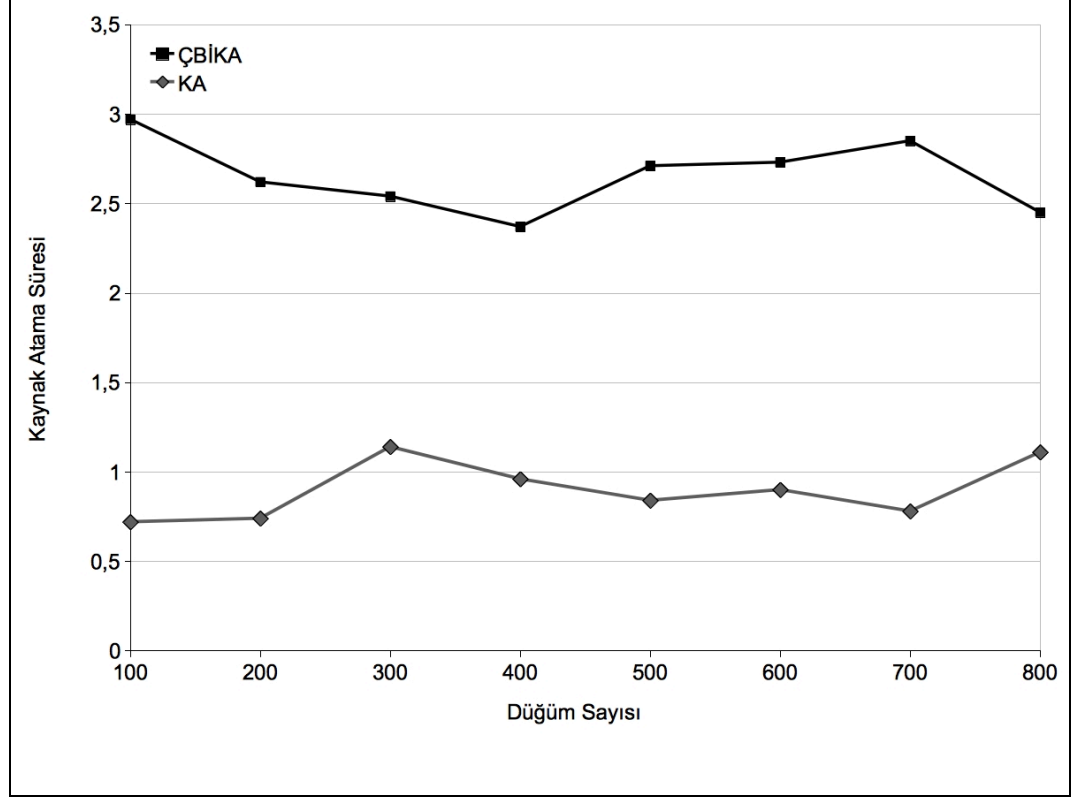
$$\{(R1 \bowtie R2) \bowtie (R3 \bowtie R4)\}$$

Şekil 4.5 Örnek sorgunun biçimsel gösterimi

Deney senaryolarında her bir taban ilişkisi yatay olarak 5 bölüme ayrılmıştır ve her bir bölüm örgü sistemi üzerinde rastgele bir düğüm üzerine yerleştirilmiştir. Bölümlerin büyüklükleri 50 GByte olacak şekilde belirlenmiştir.

Şekil 4.6 örnek sorgu için gerçekleştirilen kaynak atama süresini göstermektedir. Burada görülebileceği üzere ÇBİKA algoritması kaynak atama işlemini KA algoritmasına göre daha uzun sürede tamamlamaktadır. Bu fark, ÇBİKA algoritmasının tüm veri kaynakları ile aday düğümler arasındaki veri iletişim maliyetlerini ölçmesinden kaynaklanmaktadır. Buna karşılık KA algoritması herhangi bir ölçüm yapmamakta ve tahminlenen başarımlı artmaması halinde sonlanmaktadır. Bu sebeple hem daha iyi başarımlı sağlayabilecek düğümler irdelenmemekte hem de ataması gerçekleştirilen düğümlerin veri kaynaklarına olan uzaklıkları irdelenmemektedir. Ancak şekilde görülebileceği

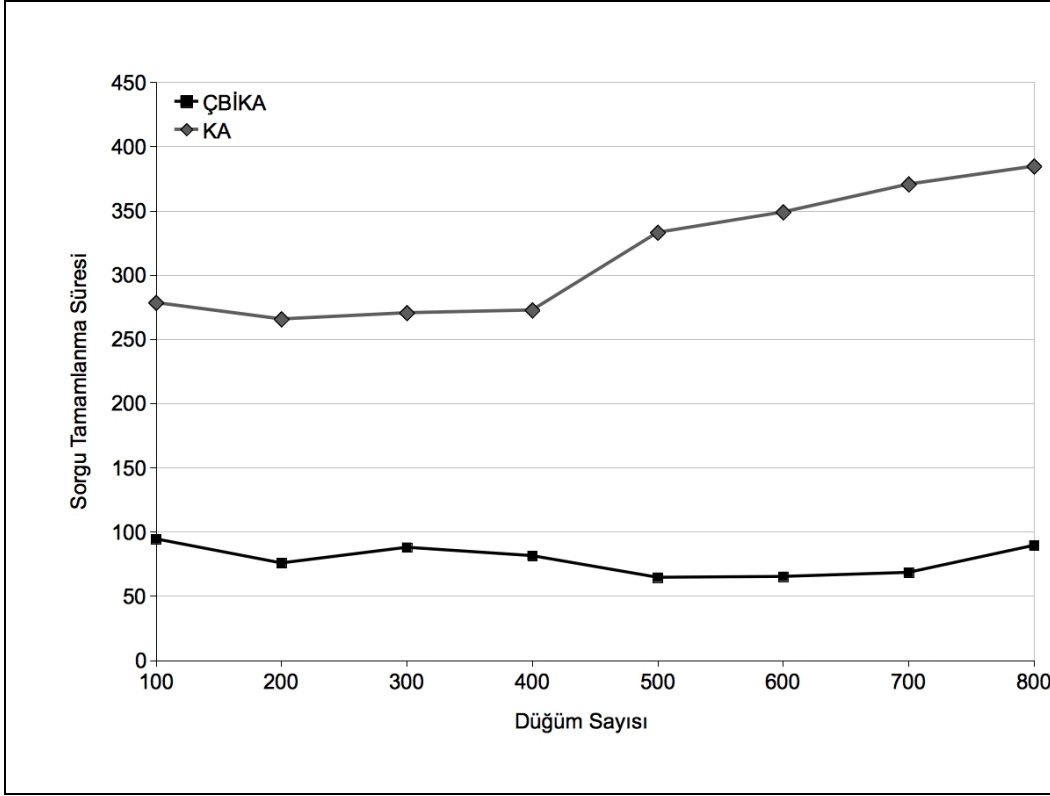
üzere ÇBİKA algoritması sistemdeki düğüm sayısındaki artıştan bağımsız şekilde sabite yakın bir başarımla sergilemiştir. Bu da algoritmanın ölçeklenebilir olduğu kuramını desteklemektedir.



Şekil 4.6 Kaynak ataması için gereken süreler

Şekil 4.7 deneylerde kullanılan örnek sorgunun, ataması tamamlanan kaynaklar üzerindeki çalışma süresini göstermektedir. Şekilde ÇBİKA algoritmasının KA algoritmasına göre daha iyi bir başarımla sergilediği görülmektedir. Bunun sebebi, ÇBİKA algoritmasının sorgu işlemesi sırasında oluşacak iletişim maliyetlerini azaltacak şekilde kaynaklar seçmesi olarak açıklanmaktadır. Buna karşılık, KA algoritması tarafından atanan kaynaklar veri kaynaklarına uzak olabilmekte, bu da sorgu işleminin başarımını olumsuz etkileyebilmektedir.

Şekil 4.6 ve 4.7’de gösterilen deney sonuçları dikkate alındığında, ÇBİKA algoritmasının başlangıç kaynak atama şeklini bulmak için harcanan ek maliyete karşılık sorgu işleminin başarımını yükselttiği sonucuna varılmaktadır. Bu durumda, uzun süreli çalışacak olan sorguların kaynak atamasında ÇBİKA algoritmasının tercih edilebilir olduğu sonucuna varılmaktadır.



Şekil 4.7 Örnek sorgu çalışma süreleri

4.3.4 Sonuç

Bu bölümde TBİKA algoritmasını kullanan, birden fazla birleştirme işletmeninden oluşan sorgulara kaynak ataması gerçekleştiren ÇBİKA algoritması önerilmiştir. Algoritma detaylı olarak anlatılmış ve analizleri sunulmuştur. Sunulan analizler ns2 benzetim ortamında gerçekleştirilen deneylerle güçlendirilmiştir. Gerçekleştirilen deneylerde ÇBİKA algoritmasının yanı sıra literatürdeki algoritmaların ortak özelliklerini taşıyan bir karşılaştırma algoritması da uygulanmış ve ÇBİKA algoritması ile karşılaştırılmaları sunulmuştur. Gerçekleştirilen deney sonuçları dikkate alındığında, uzun süreli sorgularda ÇBİKA algoritmasının benzer algoritmalarla kıyasla başarımının daha yüksek olduğu görülmüştür. Bu gibi durumlarda ÇBİKA algoritmasının tercih edilebilir olduğu sonucuna varılmıştır.

4.4 Genel Sonuçlar

Kaynak ataması, örgü sistemlerde sorgu işleminin eniyilemesi için anahtar rol alan adımlardan biri olarak tanımlanmaktadır. İşlenecek sorgu için doğru olan düğümlerin atanması sorgu işleminin başarımını doğrudan

etkilemektedir. Ancak en iyi kaynak atama şeklinin bulunması NP-Complete bir problem olarak tanımlanmıştır. Bu sebeple literatürde kaynak ataması için sezgisel yöntemler önerilmiştir. Bölüm 2’de kaynak atama algoritmaları üzerine sunulan literatür taraması çalışması sonuçları dikkate alındığında mevcut çalışmaların ortak eksik noktaları olduğu tespit edilmiştir. Öncelikle mevcut çalışmalarda irdelenen aday düğüm örnek uzayının örgü sistemindeki bütün düğümleri kapsadığı görülmüştür. Bu durum mevcut çalışmaların ölçeklenebilirliği üzerinde olumsuz etki yaratmaktadır. Bunun yanı sıra, literatürde bilgimiz dâhilinde, sorguların işletilmesi sırasında oluşacak veri iletişim maliyetlerini azaltmaya odaklı çok az çalışmaya rastlanmıştır. Bu gözlemler ışığında bu bölümde örgü sistemleri üzerinde sorgu işleme için ölçeklenebilir ve etkin kaynak atama algoritmaları önerilmiştir. Bu amaçla, veri kaynaklarına yakın düğümlerden oluşan sonlu sayıda bir aday düğüm listesi oluşturulmuş, bu liste içerisindeki düğümlerden, sorgu işlemeyi eniyileyecek olanların seçilmesi sağlanmıştır. Bu amaçla kısım 4.2’de tek bir birleştirme işletmeninden oluşan sorguların kaynak ataması için TBİKA, kısım 4.3’te ise birden çok birleştirme işletmeninden oluşan sorgulara kaynak ataması gerçekleştiren ÇBİKA algoritmaları önerilmiştir. Algoritmaların karmaşıklık analizleri sunulmuş, ns2 benzetim ortamında gerçekleştirilen deneylerde gözlemlenen çıktılar ile analizler güçlendirilmiştir. Önerilen algoritmalar literatürdeki benzer çalışmalar ile karşılaştırılmış ve benzetim ortamındaki deney sonuçlarına göre güçlü ve zayıf oldukları yönler belirtilmiştir. Buna göre, işletilecek olan sorgunun uzun süreli bir sorgu olması halinde önerilen kaynak atama algoritmalarının tercih edilebilir olduğu gösterilmiştir.

5. ÖRGÜ SİSTEMLERİNDE SORGU İŞLEME İÇİN HATAYA DAYANIKLI KAYNAK ATAMASI

Bu bölümde örgü sistemlerde sorgu işlemesi için hataya dayanıklı bir kaynak atama algoritması önerilmektedir. Önerilen algoritma durumlu işletmenlerin edilgen yinelenmeleri fikrine dayanmaktadır. Önerilen algoritmanın kuramsal analizleri sunulmakta ve benzetim ortamında gerçekleştirilen deneylerle başarımı değerlendirilmektedir.

5.1 Giriş

Örgü sistemleri, dağıtık ve koştur sistemlerden geniş ölçeklilik, devingenlik ve türdeş olmayan kaynaklara sahip olma özellikleri ile ayrılmaktadır (Foster and Kesselman, 2004). Örgü sistemlerinin bu özellikleri, dağıtık sorgu işlemesi alanında birçok yeni araştırma konusunun ortaya çıkmasına sebep olmaktadır. Bu problemlerden öne çıkanları kaynak taraması, kaynak ataması, kaynak seçimi, veri yönetimi, süreç gözlemlene, güvenlik konuları olarak gösterilmektedir (Gounaris, et al., 2005). Önceki bölümlerde kaynak tarama ve kaynak atama konuları için yeni algoritmalar önerilmiştir. Ancak sorguların örgü sistemleri gibi devingen sistemlerde etkin biçimde işletilebilmesi için bu adımlar yeterli olmamaktadır. Sistemdeki düğümlerin ortamdan habersiz bir şekilde ayrılmalrı halinde izlenecek yolun belirlenmesi, kaynak ataması adımı içerisinde incelenmesi gereken bir konudur. Örgü sistemleri devingen olmaları sebebiyle düğümlerin sistemden habersiz ayrılmalrı olağan bir durum olarak kabul edilmektedir. Bu gibi düğüm hataları sorgunun çalışmasına katılan düğümlerde gerçekleştiğinde sorgunun yeni baştan çalıştırılması yüksek maliyetli olabilmektedir. Bu sebeple örgü sistemlerde sorgu işlemesinde hataya dayanıklılık çok önemli bir konu olarak irdelenmektedir.

Literatürde devingenliği irdemiş birçok kaynak atama çalışması bulunmaktadır (Gounaris, et al., 2005; Silva, et al., 2006; Venugopal, et al., 2006; Kotowski, et al., 2008; Gounaris, et al., 2009; Paton, et al., 2009; Quiane-Ruiz, et al., 2009). Ancak bu çalışmalar düğümlerin başarımlarının devingenliğini irdemiş, durumlu işletmenlerin çalışması sırasında gerçekleşen düğüm hatalarını ve düğümlerin sistemden ayrılmalrını irdememiştir. Kodlayarak birleştirme gibi durumlu işletmenlerin durumlarının düğüm hatası sonucu kurtarılması gerektiğinden yukarıdaki devingen kaynak atama algoritmaları bu gibi durumlarda uygun olmayabilir. Bilgimiz dahilinde örgü sistemler üzerinde hataya dayanıklı

sorgu işleme sunan az sayıda çalışma bulunmaktadır (Smith and Watson, 2005; Smith and Watson, 2007; Taylor, 2008; Bestehorn, et al., 2010). Her ne kadar bu çalışmalar başarılı çözümler sunsalar da çalışmalardan hiçbirisi durumlu işletmenlerin çalıştırılması üzerinde özelleşmemiştir.

Tezin bu bölümünde durumlu işletmenlerin edilgen yinelenmesi üzerine hataya dayanıklı sorgu işleme için bir kaynak atama algoritması (*HDKA*) önerilmektedir. *HDKA* öncesi başlangıç kaynak ataması için bölüm 4'te önerilen ÇBİKA algoritması kullanılmaktadır. Başlangıç kaynak ataması tamamlandıktan sonra hataya dayanıklı sorgu işleme için edilgen yineme tabanlı bir yöntem önerilmektedir (*HDKA*). Bunun için başlangıç kaynak atamasını takiben sorgu işlemeye katılan her düğüm, üzerinde çalışacak işletmenin cinsine göre bir yineme ilkesi uygulayacaktır.

Tezin bu bölümünün bilime katkısı örgü sistemlerde sorgular içerisindeki durumlu işletmenlerin işletilmesinde kullanılacak bir hataya dayanıklılık algoritmasıdır. Bu şekilde örgü sistemlerinin geniş ölçeklilik, türdeş olmayan kaynaklar ve devingenlik özelliklerinin tümünün kapsandığı bir kaynak atama yönteminin oluşturulması hedeflenmektedir.

Tezin bu bölümü boyunca durumlu işletmenlere örnek olarak bir veya daha çok birleştirme işletmeni içeren kodlayarak birleştirme sorguları ele alınmıştır. En iyilenmiş sorgu işletmen ağacının dışarıdan verildiği varsayılmaktadır. Sorguya katılan tabloların veri kaynakları üzerinde kopyalanmadan yatay olarak parçalandığı kabul edilmektedir. Bu durumda her bir parça yalnız bir düğüm üzerinde bulunacaktır. Sorgunun gönderildiği düğümün ve veri kaynaklarının sorgunun çalışması boyunca hatalı çalışmayacakları veya kendi içlerinde hataya dayanıklı oldukları varsayılmaktadır. Sorguların birleştirme işletmenlerinden oluştuğu ve her bir birleştirme işletmeninin *tarama*, *inşa* ve *yoklama* adında görevlerden oluştukları kabul edilmektedir. *Tarama* görevleri veri kaynaklarından veri okuyup *inşa* ve *yoklama* görevlerine göndermekle yükümlüdür. *İnşa* görevleri *tarama* görevlerinden veri alıp kodlama tablosunu oluşturmakla yükümlüdür. *Yoklama* görevleri bağlı oldukları *inşa* görevleri tamamlanana kadar beklemekte daha sonra kendine bağlı *tarama* görevlerinden veri almaktadır. Alınan veri kodlama tablosunda taranarak eşleşme bulunduğu takdirde birleştirme işlemi yapılmakta ve ara sonuçlar bir sonraki birleştirme işletmenine gönderilmektedir. Birleştirme işletmenlerinden oluşan sorguların örgü sistemlerinde işlenmesi ile ilgili detaylı bilgi (Özsu and Valduries, 2011) çalışmasında gösterilmektedir.

Bu bölümün geri kalanı şu şekilde yapılanmıştır: kısım 5.2’de hataya dayanıklı kaynak atama (HDKA) algoritması önerilmektedir. Kısım 5.3’te önerilen algoritmanın analizleri sunulmaktadır. Kısım 5.4’te HDKA algoritması ns2 benzetim ortamında gerçekleştirilen deneyler ile değerlendirilmektedir. Son olarak kısım 5.5’te sonuçlar sunulmaktadır.

5.2 Hataya Dayanıklı Kaynak Atama (HDKA) Algoritması

Bu kısımda örgü sistemleri üzerinde sorgu işleme içi, durumlu birleştirme işletmenlerindeki görevlerin edilgen yinelenmeleri tabanlı bir hataya dayanıklı kaynak atama (*HDKA*) algoritması önerilmektedir. *HDKA* için sorgunun yapıldığı düğümün ve tablo bölümlerinin bulunduğu düğümlerin sorgu işlemesi boyunca hatasız çalıştığı veya hâlihazırda hataya dayanıklı oldukları varsayılmaktadır. Buna istinaden, *HDA* birleştirme görevlerinin atandığı düğümlerin hataya dayanıklılıklarından sorumludur. Bu bölüm boyunca, sorguların işletildiği düğümler *ana düğümler*, hataya dayanıklılık için kullanılan düğümler ise *yineleme düğümleri* olarak adlandırılacaktır. *Ana düğümlerin* ve *yineleme düğümlerinin* aynı anda hataya düşmedikleri varsayılmaktadır.

5.2.1 Algoritma

HDKA 4 ana adımdan oluşmaktadır, i) yineleme düğümü seçimi, ii) sorgunun çalıştırılması ve yineleme, iii) hata sezgileme ve iv) hatadan kurtarma. Bu adımlar aşağıda anlatılmaktadır:

i) *Yineleme Düğümü Seçimi*: *Yineleme düğümü* seçimi adımı *ana düğüm* çalışmaya başlamadan önce gerçekleştirilmektedir. *ÇBİKA* algoritması tamamlandıktan sonra seçilen düğümlerin her biri kendisine en yakın komşusunu *yineleme düğümü* olarak atamaktadır. En yakın komşunun seçimi *HDKA* süresince oluşacak ek yükü en aza indirmeyi amaçlamaktadır. Bu adımda *ana düğümün* her zaman yineleme amaçlı kullanılabilir en az bir komşusu olduğu varsayılmaktadır.

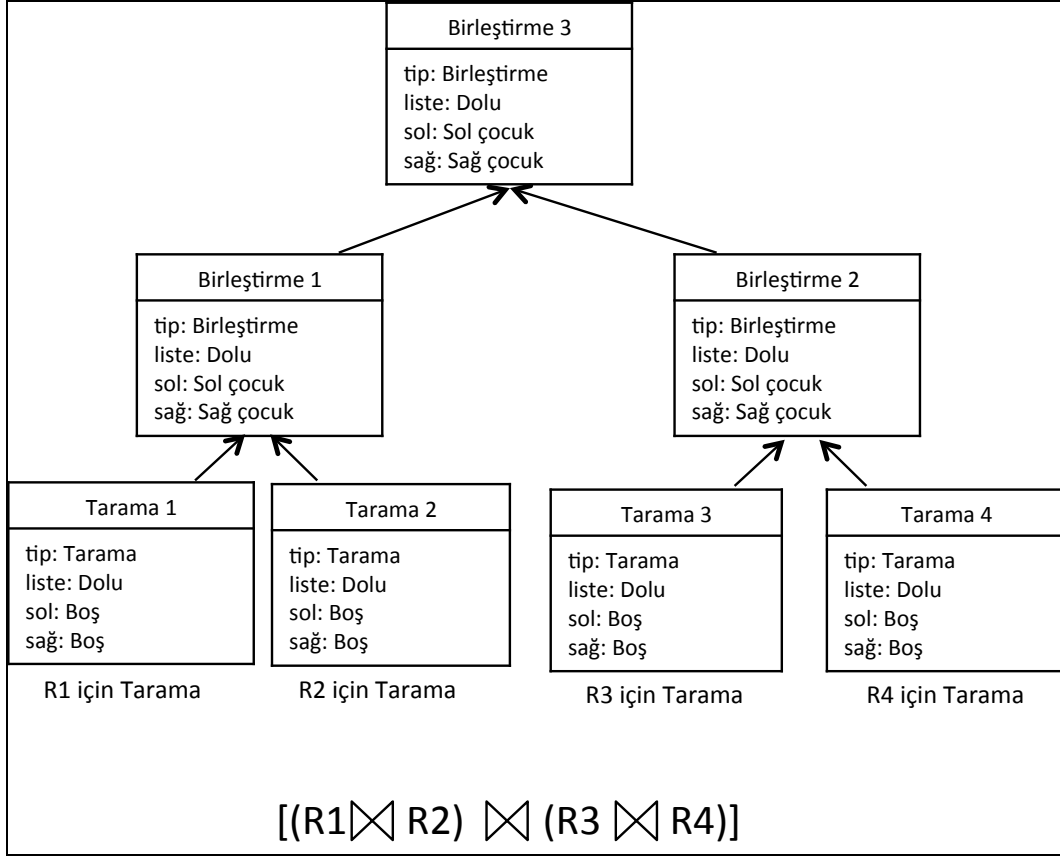
ii) *Sorgunun çalıştırılması ve yineleme*: *Yineleme düğümü* seçimi adımından sonra *ana düğüm* sorguyu çalıştırmaya başlamaktadır. Bu sırada belirli aralıklarla kendi durumunu *yineleme düğümüne* göndermektedir. Koşut çalışan *ana düğümler* ve *yineleme düğümleri* arasındaki eş zamanlama maliyetinden kaçınmak amacıyla bu adımda edilgen yineleme kullanılması uygun bulunmuştur. Yineleme sırasında yedeklenen *ana düğüm* durumları çalışmakta olan görevin tipine göre

farklılık göstermektedir. Öyle ki, *İnşa* görevi sırasında kodlama tablosu oluşturulmaktadır, bu sebeple *inşa* görevi boyunca oluşmakta olan kodlama tablosunun ve alınan en son veri paket numarasının durum olarak yedeklenmesi gerekmektedir. Öte yandan, *yoklama* görevi çalışmaya başladığında kodlama tablosu tamamlanmış ve hâlihazırda yedeklenmiş durumdadır. Bu sebeple *yoklama* görevi süresince yalnızca en son alınan veri paketi numarasının durum olarak yedeklenmesi yeterlidir. *Ana düğümlerin* durumları artımlı olarak yedeklenmektedir, her bir yedeklemede en son yedeğin üzerine eklenen durum *yineleme düğümüne* gönderilmektedir.

iii) *Hata sezgileme*: HDKA'da hata sezgileme hem *ana düğümler* hem de *yineleme düğümleri* tarafından gerçekleştirilmektedir. *Ana düğümler yineleme düğümlerini*, *yineleme düğümleri ise ana düğümleri* periyodik yineleme mesajları ve onların doğrulama mesajlarının ulaşım raporları aracılığı ile takip etmektedir.

iv) *Hatadan kurtarma*: Bir *yineleme düğümünün* hata sezgilemesi durumunda *yineleme düğümü* kendini *ana düğümü* ile yer değiştirmektedir ve bu değişikliği *ana düğümün* iletişim halinde olduğu düğümlere bildirmektedir. Bu durumdaki bir *yineleme düğümü* ana düğüm haline gelmeden önce kendine bir *yineleme düğümü* seçmektedir. Öte yandan, eğer bir *ana düğüm*, *yineleme düğümünün* hatasını sezerse, yeni bir *yineleme düğümü* seçer ve çalışmasına devam eder.

HDKA algoritması ÖBİKA algoritması ile ortaklaşa çalışmaktadır. ÇBİKA algoritması tarafından başlangıç ataması gerçekleştirilen düğümler sorgu ağacı içerisine kaydedilmektedir. İçerisinde başlangıç kaynak ataması sonuçlarının saklı tutulduğu bu sorgu ağacı HDKA algoritmasına girdi olarak verilmektedir. Örnek bir sorgu ağacı şekil 5.1'de gösterilmektedir.



Şekil 5.1 HDKA Algoritmasına girdi olarak sunulan sorgu ağaç gösterimi

Sorgu ağacı üzerindeki her bir köşe içerisinde *görevElemanı* adında bir veri yapısı bulunmaktadır. Bu veri yapısının içeriği Şekil 5.2’de gösterilmektedir. *görevElemanı* veri yapısı içerisinde *görevTipi*, *düğümListesi*, *sağİşaretçi* ve *solİşaretçi* olmak üzere 4 adet alan bulunmaktadır. *görevTipi* alanı sorgu ağacı üzerindeki köşenin görev tipini (*tarama veya birleştirme görevi*) göstermektedir. HDKA algoritmasının başlangıcında, sorgu ağacının yapraklarındaki düğümlerde *düğümListesi* alanı görev için atanmış olan düğümlerin listesini tutmaktadır. *sağİşaretçi* ve *solİşaretçi* alanları ise köşenin sağ ve sol çocuklarını işaret etmektedir.

görevElemanı
• <i>integer</i> : görevTipi
• <i>array</i> : adayListesi
• <i>pointer</i> : solİşaretçi
• <i>pointer</i> : sağİşaretçi

Şekil 5.2 Sorgu ağacının köşelerini temsil eden veri yapısı

HDKA algoritması sorgu ağacını girdi olarak kabul etmekte ve sondan başlayarak gezinme ile işleme almaktadır. Algoritmanın çalışması sırasında her düğüm ziyaretinde düğümün türüne göre bir yedekleme şekli uygulanmaktadır. HDKA algoritmasının sorgunun gönderildiği düğümde çalışmasının biçimsel gösterimi algoritma 5.1’de gösterilmektedir.

Algoritma 5.1. Sorgunun gönderildiği düğümde HDKA algoritması

Girdi: *Sorgu ağacının kökü*

Çıktı: *Sorgunun işlenmesi*

- 1: *HDKAAgoritması(ağaçDüğümü)*
 - 2: **eğer** *ağaçDüğümü boş ise*
 - 3: *dönüş*
 - 4: **aksi halde**
 - 5: *HDKAAgoritması(ağaçDüğümü.sol)*
 - 6: *HDKAAgoritması(ağaçDüğümü.sağ)*
 - 7: **eğer** *ağaçDüğümü.tip = TARAMA ise*
 - 8: *ağaçDüğümü.liste içerisindeki düğümlere AllocScanReq mesajı gönder*
 - 9: **aksi halde** *ağaçDüğümü.tip = BİRLEŞTİRME ise*
 - 10: *ağaçDüğümü.liste içerisindeki düğümlere AllocJoinReq mesajı gönder*
 - 11: **son**
 - 12: **son**
 - 13: *dönüş*
 - 14: **son**
-

Sorgu ağacı gezinme işleminin tamamlanmasının ardından sorgulanan düğüm sorgunun sonuçlarını almak üzere beklemeye geçmektedir. Örgü sistemi üzerindeki, sorgu işlemede kullanılacak olan diğer düğümler, sorgulanan düğüm tarafından gönderilen mesajları aldıktan sonra şekil 5.4’te gösterilen sonlu konum makinesinde gösterilen adımları yürütmektedirler. Farklı türdeki görevlere atanmış olan düğümlerde işletilen algoritmalar ana hatlarıyla aşağıda gösterilmektedir. Algoritma 5.2’de HDKA algoritmasının bir tarama düğümü

üzerindeki çalışması gösterilmektedir. Bir birleştirme düğümü üzerinde HDKA algoritmasının çalışması algoritma 5.3'te gösterilmektedir. Son olarak, algoritma 5.4'te yedekleme amacıyla kullanılacak düğümler üzerinde HDKA algoritmasının çalışması gösterilmektedir.

Algoritma 5.2. Tarama düğümü üzerinde HDKA algoritması

Girdi: Alınan mesaj

Çıktı: Gerekli adımların yürütülmesi

- 1: *AllocScanReq* mesajı alınması halinde:
 - 2: *Tüm ebeveyn düğümlerden StartScan mesajı alınana kadar bekle*
 - 3: *StartScan mesajları tamamlandıktan sonra*
 - 4: *Tarama işlemini başlat*
 - 5: *Taranacak veri tükendiğinde*
 - 6: *Tüm ebeveyn düğümlere ScanEnd mesajı gönder*
 - 7: **son**
-

Algoritma 5.3. Birleştirme düğümleri üzerinde HDKA algoritması

Girdi: Alınan mesaj

Çıktı: Gerekli adımların yürütülmesi

- 1: *AllocJoinReq* mesajının alınmasını takiben:
 - 2: *En yakın komşuya ReplicaReq mesajı gönder*
 - 3: *ReplicaResp mesajının alınmasını takiben tarama düğümlere StartScan mesajı gönder*
 - 4: *Alınan tarama verileri ile kodlama tablosunu oluştur*
 - 5: *Yedekleme düğümüne dönemsel olarak kodlama tablosundaki değişiklikleri gönder*
 - 6: **eğer** yedekleme düğümü ayrılırsa
 - 7: *Başka bir yedekleme düğümü belirle*
 - 8: **son**
 - 9: *İnşa görevinin tamamlanmasını takiben yoklama ilişkisine ait tarama düğümlerine StartScan mesajı gönder*
-

-
- 10: *Alınan tarama verisini kullanarak yoklama işlemini yürüt*
 - 11: *Dönemsel olarak en son alınan tarama mesajının seri numarasını yedekleme düğümüne bildir*
 - 12: *Yoklama işleminin bitmesinin ardından ebeveyn düğümlere ScanEnd mesajı gönder*
 - 13: **son**
-

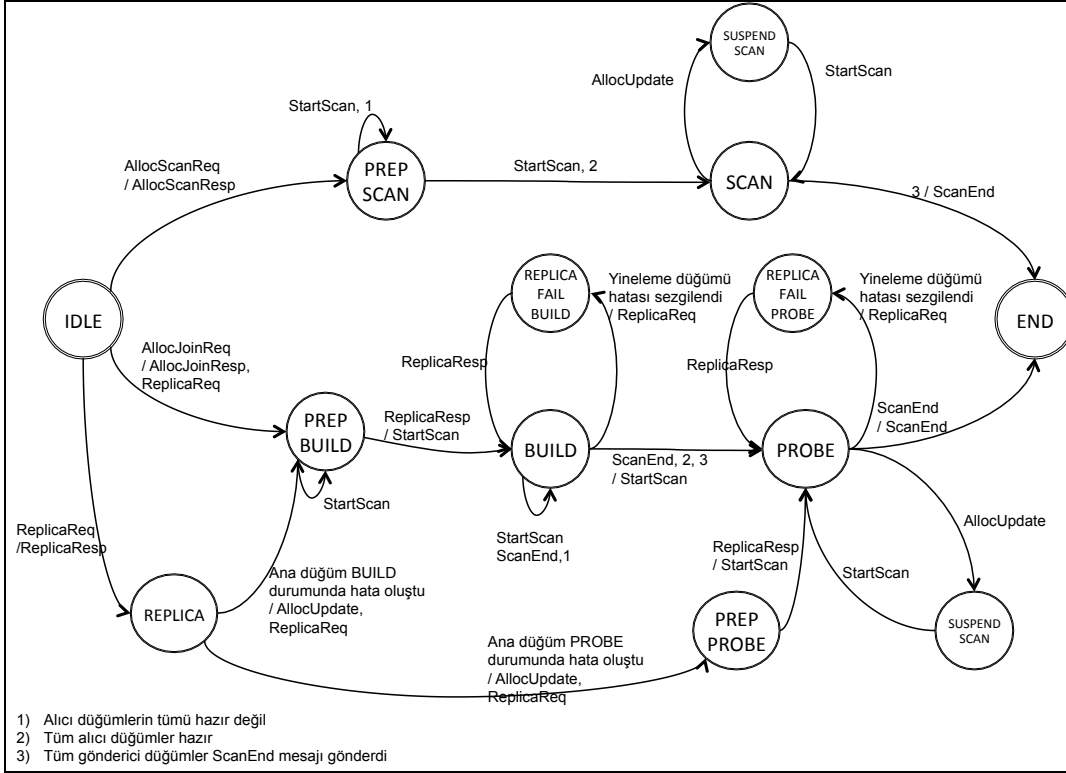
Algoritma 5.4. Yedekleme düğümlerinde HDKA algoritmasının çalışması

Girdi: *Alınan mesaj*

Çıktı: *Gerekli işlemlerin yürütülmesi*

- 1: *ReplicaReq mesajının alınmasını takiben:*
 - 2: *Gönderici düğüme ReplicaResp mesajı gönder*
 - 3: *Durum güncellemeleri alındıkça kaydet*
 - 4: **eğer** *ana düğüm ayrılırsa*
 - 5: *Ana düğüme ait tarama düğümlerinin tümüne iletişimin askıya alınması için AllocUpdate mesajı gönder*
 - 6: *En yakın komşuya ReplicaReq mesajı gönder*
 - 7: *Görev tipini birleştirme olarak güncelle*
 - 8: *Ana düğüme ait tüm tarama düğümlerine StartScan mesajı göndererek iletişimi kendi üzerinden devam ettir*
 - 9: **son**
 - 10: **son**
-

HDKA algoritmasının detaylı anlatımının gösterildiği sonlu durum makinesi şekil 5.3'te gösterilmektedir.



Şekil 5.3 HDKA Algoritmasının sonlu durum makinesi

HDKA'da kullanılan mesaj tipleri ve açıklamaları aşağıdaki gibidir:

- *AllocScanReq*: Bu mesaj sorgunun yapıldığı düğüm tarafından tablo bölümlerinin bulunduğu düğümlere gönderilmektedir. Bu şekilde bu düğümlere *tarama* görevi için seçildikleri bildirilmektedir.
- *AllocScanResp*: Veri kaynakları *tarama* görevi ataması için gerçekleştirilen bildirimi bu mesaj ile onaylamaktadır.
- *AllocJoinReq*: Birleştirme görevi için seçilmiş olan düğümlerin atama işlemi bu mesaj ile gerçekleştirilmektedir.
- *AllocJoinResp*: Birleştirme görevi için atama istemi alan düğümler bu mesaj ile cevap vermektedir.
- *ReplicaReq*: Birleştirme görevi için ataması yapılan düğümler çalışmaya başlamadan önce müsait durumdaki en yakın komşusuna *yineleme düğümü* olması için bu mesajı göndererek istemde bulunmaktadırlar.

- *ReplicaResp*: Yineleme istemi alan bir düğüm uygun olması halinde bu mesaj ile yineleme düğümü olmasını teyit etmektedir.

- *StartScan*: Birleştirme görevi için atanan bir düğüm *yineleme düğümünü* seçtikten sonra sorguyu çalıştırmaya hazır hale gelir ve kendisine veri gönderecek olan *tarama düğümlerine* bu mesajı göndererek veri alımına hazır olduğunu bildirmektedir.

- *AllocUpdate*: Sorgunun çalışması sırasında, bir *yineleme düğümü*, *ana düğümünün* hatasını sezgilediği zaman kendisi *ana düğüm* haline gelmekte ve iletişim halindeki düğümlere değişikliği bu mesaj ile bildirmektedir. Bu mesaj, aynı zamanda işlemin en son kaldığı noktanın bilgisini de içermektedir.

- *ScanEnd*: Bu mesaj gönderilmekte olan verinin sonu olduğunu ifade etmektedir.

HDKA'da her bir düğümün içinde bulunabileceği 11 adet durum bulunmaktadır. Bunlar aşağıdaki gibidir:

- *IDLE*: *HDKA'da* tüm düğümler başlangıçta bu durumdadırlar.

- *PREP_SCAN*: *AllocScanReq* tipindeki bir mesaj bir düğüm tarafından alındığı zaman düğüm durumunu *PREP_SCAN'e* geçirmektedir. Düğümler, veri gönderilecek olan tüm birleştirme düğümleri hazır oluncaya kadar bu durumda beklemektedir.

- *SCAN*: *PREP_SCAN* durumundaki bir düğüm, veri göndereceği birleştirme düğümlerinin tümü hazır olduğunda *SCAN* durumuna geçmektedir. *SCAN* durumu boyunca düğümler depolama alanından verileri okuyup ilgili birleştirme düğümüne göndermektedir.

- *SUSPEND_SCAN*: Birleştirme düğümlerinden biri hataya düştüğü zaman hatadan kurtarıma işi tamamlanıncaya kadar hatalı düğüme veri akışı bu duruma geçilerek askıya alınmaktadır.

- *PREP_BUILD*: *İnşa* görevi için seçilen bir düğüm *AllocJoinReq* mesajını almasını takiben bu duruma geçerek *yineleme düğümü* atamasının

tamamlanmasını beklemektedir.

- *BUILD*: *PREP_BUILD* durumundaki bir düğüm *yineleme düğümünün* atanmasını takiben bu duruma geçmekte ve kendinden önceki düğüm tarafından gönderilen veriyi kullanarak kodlama tablosunu oluşturmaktadır.

- *PROBE*: *BUILD* durumundaki bir düğüm kodlama tablosu tamamlandıktan sonra *PROBE* durumuna geçmektedir. Bu esnada yoklamanın yapılacağı tablonun bölümlerinin bulunduğu düğümlere *StartScan* mesajı gönderilmektedir. Bu şekilde yoklama işlemi başlamakta ve sonuçlar ilgili düğümlere gönderilmektedir.

- *REPLICA_FAIL_BUILD* ve *REPLICA_FAIL_PROBE*: Bir *yineleme düğümünün* hataya düştüğü sezgilendiği zaman ilgili *ana düğümün* durumu bu durumlardan birine çevrilmekte ve yeni bir *yineleme düğümü* atanıncaya kadar bu durumda bekletilmektedir.

- *REPLICA*: Yineleme düğümü olma istemi alan düğümler bu duruma geçerek *yineleme düğümleri* haline gelmektedir.

- *PREP_PROBE*: *PROBE* durumundaki bir düğümün hataya düşmesi halinde düğümün *yineleme düğümü* durumunu *PREP_PROBE*'a geçirmektedir ve kendine bir *yineleme düğümü* aramaktadır. *Yineleme düğümünün* bulunması ve atanmasını takiben düğüm *ana düğüm* olarak yoklama görevini sürdürmeye devam etmektedir.

- *END*: *HDA* algoritmasını tamamlayan düğümler bu duruma geçmektedirler. Algoritmaya katılan tüm düğümlerin algoritmayı bu durumda bitirmeleri beklenmektedir.

5.2.2 Analiz

Bu bölümde HDKA algoritmasının zaman ve mesaj karmaşıklıkları sunulmaktadır.

Kuram HDKA algoritmasının ÇBİKA algoritması üzerine getirdiği ek maliyet zaman karmaşıklığı açısından j birleştirme işletmeni sayısını gösterecek şekilde $O(j)$ olarak gösterilmektedir.

İspat. HDKA algoritmasında ÇBİKA algoritmasına ek olarak yedekleme düğümü seçimi adımı çalıştırılmaktadır. Her bir birleştirme işletmeni için yedekleme düğümü seçimi 4 adımda gerçekleşmektedir. Sorguda j adet birleştirme işletmeni olması halinde HDKA algoritması toplamda $(4j)$ adımda gerçekleşmektedir. Bu da büyük O gösteriminde $O(j)$ olarak gösterilmektedir.

Kuram HDKA algoritmasının ÖBİKA algoritması üzerine getirdiği mesaj karmaşıklığı j birleştirme işletmeni sayısını, n veri kaynağı sayısını ve m birleştirme işletmenlerine atanan toplam düğüm sayısını temsil edecek şekilde $O(jnm)$ olarak hesaplanmaktadır.

İspat. HDKA algoritmasında birleştirme işletmenlerine ait tarama görevleri ancak yedekleme işlemlerinin tamamlanmasından sonra başlatılabilmektedir. Bu durum birleştirme düğümlerinin tarama düğümlerine birer bilgilendirme mesajı göndermelerini gerektirmektedir. Bu sebeple m adet birleştirme düğümü toplamda n adet tarama düğümüne mesaj göndermektedir. Toplam j adet birleştirme işletmeni için gerekli olan toplam mesaj sayısı $O(jnm)$ olarak hesaplanabilmektedir.

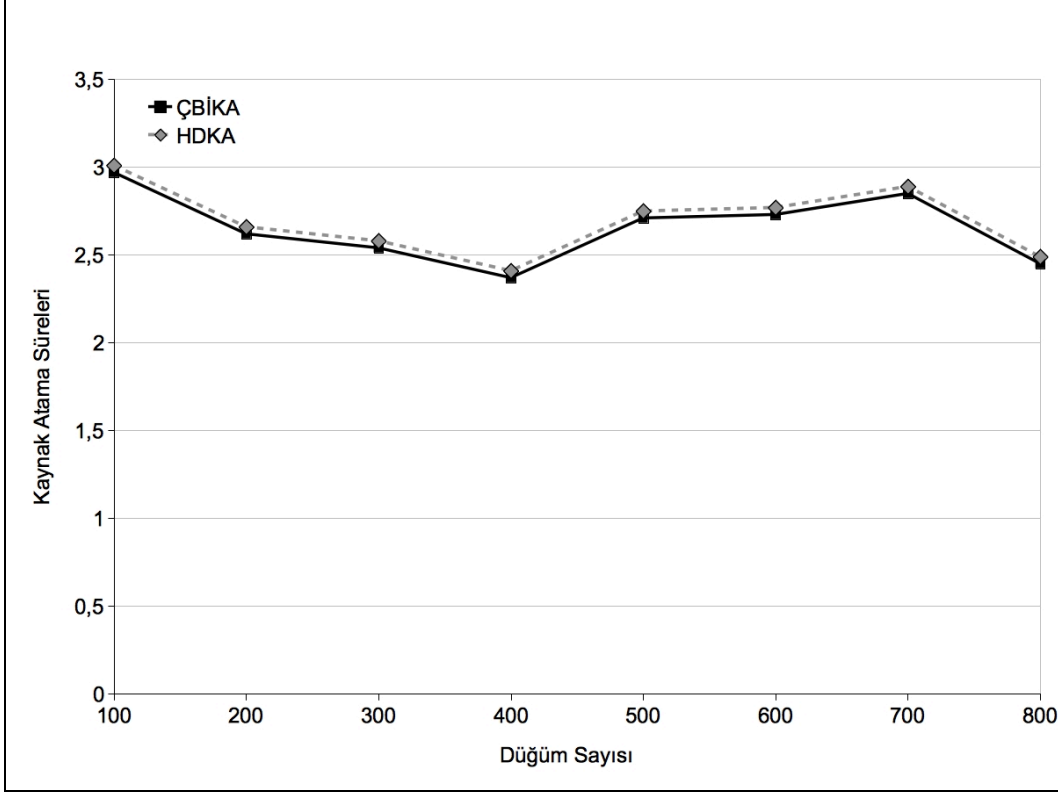
5.2.3 Benzetimler

Bu kısımda HDKA algoritması için ns2 benzetim ortamında gerçekleştirilen deneyler ile niceliksel değerlendirmeler sunulmaktadır. HDKA algoritması ÇBİKA algoritması ile karşılaştırılmış, getirdiği ek maliyet gözlemlenmiştir.

Benzetim ortamı deney senaryolarında kısım 4.3.3'te kullanılan senaryolar kullanılmaktadır. Deneylerde kaynak atama işleminin tamamlanması için gereken süre ölçülmüştür.

Şekil 5.4'te HDKA ve ÇBİKA algoritmaları için kaynak atama süreleri gösterilmektedir. Şekilden görülebileceği üzere HDKA ve ÇBİKA algoritmaları arasında az bir fark bulunmaktadır. Bu fark HDKA algoritmasının sorgu işlemeye başlamasından önce, atanan düğümler için yedekleme düğümlerinin seçilmesinden kaynaklanmaktadır. Senaryolarda birleştirme işletmeni ve veri kaynağı sayısı sabit olduğundan, HDKA algoritmasının ek maliyeti birleştirme

işletmenlerine atanan düğüm sayısına bağlı kalmaktadır. Birleştirme işletmenlerine atanan düğüm sayıları, farklı senaryolarda ortamdaki toplam düğüm sayısına kıyasla fazla değişmemesi sebebiyle deneyler boyunca ek maliyet neredeyse sabit kalmaktadır. Bu durum kuramsal analizleri doğrular niteliktedir.



Şekil 5.4 HDKA algoritması kaynak atama süreleri

5.3 Genel Sonuçlar

Örgü sistemler üzerinde sorgu işleme için kaynak ataması, sorgu işleminin başarımını doğrudan etkileyen önemli bir adımdır. Örgü sistemlerinin devingenlik, kaynakların türdeş olmaması ve geniş ölçeklilik özellikleri ile diğer dağıtık sistemlerden ayrılması sebebiyle bu ortamlar için önerilen kaynak atama algoritmasının tüm bu özellikleri karşılayabilmesi gerekmektedir. Bölüm 4'te önerilen ÇBİKA algoritması ile, örgü sistemlerinin geniş ölçeklilik ve kaynakların türdeş olmaması özellikleri karşılanmaktaydı.

Literatürde örgü sistemlerinin devingenliğini irdeleyen birçok kaynak atama algoritması bulunmasına rağmen, düğümlerin sistemden ayrılmaları durumunu irdeleyen, özellikle örgü sistemleri üzerinde sorgu işlemeye odaklı çalışmalar bilimiz dâhilinde bulunmamaktadır. Bu sebeple tezimizin bu bölümünde, örgü

sistemlerinin devingenlik özelliğinin karşılanması amacıyla hataya dayanıklı kaynak atama algoritması (*HDKA*) önerilmiştir.

Önerilen HDKA algoritması detaylı olarak anlatılmış, karmaşıklık analizleri sunulmuştur. Algoritmanın teorik analizleri, ns2 benzetim ortamında gerçekleştirilen deney sonuçları ile güçlendirilmiştir. Deney sonuçları ile hataya dayanıklılık için ÇBİKA algoritmasının üzerine getirilen ek maliyetler gösterilmiştir. Benzetim ortamı deney sonuçları HDKA algoritmasının ÇBİKA algoritması üzerine getirdiği ek maliyetin görmezden gelinebileceği sonucuna varılmış, bu sebeple, düğümlerin sistemden ayrılma durumlarının sık karşılaşıldığı ortamlarda ÇBİKA algoritmasının yerine tercih edilebilir bir yöntem sunduğu anlaşılmıştır.

6. SONUÇLAR

Bu bölümde, bu tez dâhilinde hazırlanan çalışmaların özeti sunulmakta ve önerilen her bir çalışmanın bilime katkısı açıklanmaktadır. Önerilen çalışmaların getirdiği yeni araştırma alanları ve gelecekte yapılması planlanan çalışmalar tartışılmaktadır. Son olarak bu tezin bütünü göz önüne alınarak varılan sonuçlar sunulmaktadır.

6.1 Çalışmaların Özeti

Kaynak taraması ve kaynak ataması, örgü sistemlerde sorgu işleme konusunda irdelenmesi gereken en önemli konular arasında gösterilmektedir. Bu tezde önerilen çalışmalar ile kaynak taraması ve kaynak ataması problemlerine, örgü sistemlerinin özellikleri ve sorguların gereksinimleri dikkate alınarak çözümler önerilmektedir. Bu kapsamda algoritmalar önerilmiş, önerilen her bir algoritma için kuramsal analizler sunulmuş ve bu analizlerin doğruluğu benzetim ortamında gerçekleştirilen deneyler ile gösterilmiştir. Önerilen algoritmalar benzetim ortamında gerçekleştirilen deneyler aracılığı ile literatürde bulunan benzer çalışmalar ile karşılaştırılmıştır. Karşılaştırmalar sonucunda elde edilen veriler tartışılmış, önerilen algoritmaların güçlü ve zayıf yönleri gösterilmiştir. Elde edilen sonuçlar dikkate alınarak önerilen algoritmaların fayda kazanımı sağlayabileceği durumlar açıklanmıştır. Bu tezde sunulan çalışmaların bilime katkıları aşağıda listelenmektedir:

i) *Literatür taraması:* Bölüm 2’de detaylı bir literatür taraması sunulmuştur. Her bir araştırma alanı için değerlendirme ölçütleri belirlenmiştir. Araştırma alanları için literatürdeki güncel çalışmalar sınıflandırılarak detaylı olarak incelenmiştir. İncelenen çalışmaların ortak eksiklikleri belirlenmiş ve her bir araştırma alanı için olası katkı noktaları listelenmiştir. Bölüm 2’de sunulan literatür taramasının kısımları (Cokuslu, et al., 2009; Cokuslu, et al., 2010b; Cokuslu, et al., 2012a; Hameurlain, et al., 2010) çalışmalarında yayınlanmıştır.

ii) *Kaynak taraması:* Literatürdeki mevcut çalışmalar incelendikten sonra elde edilen olası katkı noktaları irdelenmiştir. Buna göre örgü sistemlerde sorgu işleme için kullanılacak olan kaynak tarama yönteminin bir ilinge denetimi ile ortak çalışması gerektiği sonucuna varılmıştır. Bu sebeple öncelikle amaca uygun olacak şekilde bir ilinge denetimi önerilmiştir. Bunun

için, örgü sistemlerinin özellikleri ve sorgu işleminin gereksinimleri gözetilerek, bölüm 3'te öz kararlı kapsayan ağaç yapısı oluşturan üç adet yeni algoritma önerilmiştir. Önerilen algoritmalarda kapsayan ağaçların çaplarının küçük olması hedeflenmiştir. Bu sebeple algoritmalar derecesi en yüksek düğüm ağaç kökü olacak şekilde tasarlanmıştır. Önerilen ilinge denetimi ile örgü sistemlerin devingenlik ve geniş ölçeklilik özellikleri karşılanmıştır. Algoritmalar ns2 benzetim ortamında uygulanmış ve benzer çalışmalar ile karşılaştırılmıştır. Önerilen algoritmaların literatürdeki benzer çalışmalara göre daha iyi sonuçlar verdiği gösterilmiştir. Önerilen ilinge denetiminden sonra, oluşturulan kapsayan ağaç yapısını kullanarak kaynak taraması işlemini gerçekleştiren kapsayan ağaç tabanlı kaynak tarama (KAKT) algoritması önerilmiştir. Önerilen algoritma kaynak taraması sırasında sisteme gönderilecek mesajların iletimi için kapsayan ağaç yapısını kullanmaktadır. Literatürde KAKT algoritmasına benzer özelliklerde tasarlanan algoritmaların büyük çoğunluğunun kaynak taramasında kullanılan mesajların dağıtılması için taşıma yöntemini kullandıkları saptanmıştır. Bu sebeple KAKT algoritması, ns2 benzetim ortamında gerçekleştirilen deneylerde taşıma yöntemini kullanan benzer algoritmalar ile karşılaştırılmıştır. Deney sonuçlarında KAKT algoritmasının karşılaştırılan çalışmalardan daha iyi sonuçlar verdiği gösterilmiştir. Bu bölümde önerilen çalışmaların bir kısmı (Cokuslu, et al., 2010a) çalışmasında yayınlanmıştır.

iii) Kaynak atama: Bölüm 4'te, kaynak tarama adımının tamamlanmasından sonra bir başka önemli adım olan örgü sistemlerde sorgu işleme için kaynak ataması problemine odaklanılmıştır. Literatürdeki mevcut kaynak atama yöntemleri ile ilgili sunulan değerlendirmeler dikkate alınarak, bölüm 4'te kaynak atamasında kullanılacak aday düğüm arama uzayının küçültülmesi ve aday düğüm listesinin oluşturulmasında veri kaynaklarına olan uzaklıkların dikkate alınması hedeflenmiştir. Bunun için öncelikle tek bir birleştirme işletmeninden oluşan sorgulara kaynak ataması yapan tekli birleştirme işletmeni kaynak atama (TBİKA) algoritması önerilmiştir. TBİKA'da aday düğüm listeleri, düğümlerin veri kaynaklarına olan uzaklıkları dikkate alınarak küçültülmekte ve elde edilen aday düğüm listesi içerisinden sorgu işlemini eniyileyecek kaynaklar atanmaktadır. TBİKA algoritmasından sonra birden çok birleştirme işletmeninden oluşan sorgulara kaynak ataması gerçekleştiren çoklu birleştirme işletmeni kaynak atama (ÇBİKA) algoritması önerilmiştir. Önerilen algoritmaların kuramsal analizleri sunulmuş, analizler benzetim ortamında gerçekleştirilen deneylerle

güçlendirilmiştir. Önerilen algoritmalar literatürdeki benzer çalışmalarla karşılaştırılmış, fayda kazanımı elde edilecek durumlar belirlenmiştir. Bu bölümde önerilen algoritmaların bir bölümü (Cokuslu, et al., 2012b) çalışmasında yayınlanmıştır.

iv) Hataya dayanıklı kaynak atama: Bölüm 4’te önerilen kaynak atama algoritmaları sorguların gereksinimlerini karşılarken örgü sistemlerinin de türdeş olmayan kaynaklar ve geniş ölçeklilik özelliklerine çözüm getirmekteydiler. Bölüm 5’te, örgü sistemlerinin devingenlik özelliğinden kaynaklanan problemlere çözüm getirilmesi hedeflenmiştir. Bölüm 2’de sunulan literatür çalışması sonuçları dikkate alındığında, literatürde düğümlerin sistemden ayrılmalari durumunu irdeleyen, özellikle örgü sistemler üzerinde sorgu işleme konusuna odaklı kaynak atama algoritmalarının bulunmadığı gözlemlenmiştir. Bu sebeple bölüm 5’te hataya dayanıklı kaynak atama (HDKA) algoritması önerilmiştir. Önerilen algoritmanın kuramsal analizleri sunulmuş, hataya dayanıklılık için getirilmiş olan ek maliyetler hesaplanmıştır. HDKA algoritması ns2 benzetim ortamında gerçekleştirilen deneyler sayesinde ÇBİKA algoritması ile karşılaştırılmış, kuramsal ek maliyetlerin doğruluğu gösterilmiş, algoritmanın kullanımının fayda kazanımı sağlayacağı durumlar incelenmiştir.

6.2 Gelecekte Yapılması Planlanan İşler

Bu kısımda, bu tezin hazırlanması sırasında karşılaşılan yeni problemler ve olası gelecek çalışma konuları sunulmaktadır.

i) Örgü sistemlerde sorgu işleme için kaynak taraması konusunda karşılaşılan olası çalışma alanları: Kısım 3.2’de kaynak taramasında kullanılmak üzere öz kararlı kapsayan ağaç oluşturma algoritmaları önerilmiştir. Önerilen algoritmalar ile daha küçük çaplı kapsayan ağaç kullanımının kaynak tarama işleminin başarımını arttırdığı gösterilmiştir. Bu amaçla, çizgenin merkezinde bulunan düğümü kök olarak kabul edecek kapsayan ağaç yapısının daha küçük çaplı bir kapsayan ağaç olacağı düşünülmektedir. Ancak literatürde bu konuda çok az çalışmaya rastlanmıştır. Bu sebeple, kaynak taramasının başarımını arttırmak amacıyla, bu konuda daha fazla çalışma yapılabileceği düşünülmektedir.

ii) *Örgü sistemlerde sorgu işleme için kaynak ataması konusunda karşılaşılan olası çalışma alanları:* Kısım 4.2’de sorgu işleminde kullanılmak üzere veri kaynaklarına daha yakın olan düğümler seçilmesi hedeflenmiştir. Bunun için, G çizgesinin elemanları arasından, bir S alt çizgesinin orta düğümünü bulan bir algoritma gerekmektedir. Bilgimiz dâhilinde literatürde böyle bir çalışma bulunamamıştır. Böyle bir çalışmanın kaynak ataması için daha iyi adaylar seçebileceği düşünülmektedir. Bölüm 4’te önerilen algoritmalar bu probleme yaklaşımsal çözümler üretmişlerdir. Önerilen algoritmaların iyileştirilerek bu konuda daha detaylı çalışmalar haline getirilebileceği düşünülmektedir. Bunun da ötesinde, bahsedilen problem için ağırlıklı orta noktaların bulunmasının ayrıca incelenmesi gereken bir konu olduğu düşünülmektedir. Ağırlıklı orta noktaların bulunmasının, farklı büyüklüklerde olan birleştirme işletmenleri için dengeli bir kaynak atama modelinin oluşturulmasına katkıda bulunacağı düşünülmektedir.

iii) *Örgü sistemlerde sorgu işleme için hataya dayanıklı kaynak ataması konusunda karşılaşılan olası çalışma alanları:* Bölüm 5’te sorgulanan düğümün ve veri kaynaklarının hatasız çalıştığı varsayılarak, sistemdeki diğer düğümler için bir hataya dayanıklı kaynak atama algoritması önerilmiştir. Örgü sistemlerinde bu varsayımın sağlanamayacağı ihtimali düşünüldüğünde bu varsayımların olmadığı bir yöntem tasarımının bu konuda ek bir katkı sağlayacağı düşünülmektedir.

Yukarıda bahsedilen çalışma alanları dikkate alınarak daha etkin kaynak tarama ve kaynak atama algoritmaları geliştirilebileceği düşünülmektedir.

6.3 Genel Sonuçlar

Bu tezde, örgü sistemlerde sorgu işleme için kaynak taraması ve kaynak ataması konularında çalışmalar önerilmiştir. Bu amaçla, öncelikle örgü sistemlerinin ayırt edici özellikleri ve sorguların gereksinimleri belirlenmiştir. Daha sonra bu özellik ve gereksinimlerden kaynaklanan problemler listelenmiştir. Hazırlanan literatür çalışmasında gerçekleştirilen değerlendirmeler sonucunda bu problemlerden henüz çözüm getirilmemiş olanları belirlenmiş, bu konulara odaklanılarak algoritmalar önerilmiştir. Bu tezde önerilen algoritmaların, örgü sistemler üzerinde sorgu işleme konusu üzerine birçok alanda katkı sunacağı

düşünülmektedir. Sunulan katkının yanı sıra birçok yeni araştırma alanının belirlenmesine yardımcı olduğuna inanılmaktadır.

KAYNAKLAR DİZİNİ

- Abadi, D., Ahmad, Y., Balazinska, M., Cetintemel, U., Cherniack, M., Hwang, J., Lindner, W., Maskey, A., Rasin, A., Ryvkina, E., Tatbul, N., Xing, Y., and Zdonik, S.**, Year, The Design of the Borealis Stream Processing Engine, in *2nd Biennial Conference on Innovative Data Systems Research (CIDR'05)*, pp. 277-289.
- Afek, Y., Kutten, S., and Yung, M.**, 1991, Memory-Efficient Self Stabilizing Protocols for General Networks, WDAG90: Proceedings of the 4th International Workshop on Distributed Algorithms.
- Ahuja, M. and Zhu, Y.**, 1989, A Distributed Algorithm for Minimum Weight Spanning Trees Based on Echo Algorithms, Proceedings of the 9th International Conference on Distributed Computing Systems.
- Alpdemir, M., Mukherjee, A., Gounaris, A., Paton, N., Watson, P., Fernandes, A., and Fitzgerald, D.**, 2004, OGSA-DQP: A Service for Distributed Querying on the Grid, Advances in Database Technology - EDBT 2004, Springer Berlin / Heidelberg, 3923-3923.
- Androutsellis-theotokis, S. and Spinellis, D.**, 2004, A survey of peer-to-peer content distribution technologies, ACM Computing Surveys, 36:335-371pp.
- Andrzejak, A. and Xu, Z.**, 2002, Scalable, efficient range queries for Grid information services, 2nd Int. Conf. on Peer-to-Peer Computing, P2P 2002.
- Antonioletti, M., Atkinson, M., Baxter, R., Borley, A., Hong, N. P. C., Collins, B., Hardman, N., Hume, A. C., Knox, A., Jackson, M., Krause, A., Laws, S., Magowan, J., Paton, N. W., Pearson, D., Sugden, T., Watson, P., and Westhead, M.**, 2005, The design and implementation of Grid database services in OGSA-DAI: Research Articles, Concurr. Comput. : Pract. Exper., 17:357-376pp.
- Antoniou, G. and Srimani, P. K.**, 1995, A Self-Stabilizing Distributed Algorithm to Construct An Arbitrary Spanning Tree of a Connected Graph, Computers and Mathematics with Applications, 30:1-7pp.
- Antoniou, G. and Srimani, P. K.**, 1997, Distributed Self-Stabilizing Algorithm for Minimum Spanning Tree Construction, Euro-Par97: Proceedings of the Third International Euro-Par Conference on Parallel Processing, London, UK.
- Awerbuch, B.**, 1987, Optimal Distributed Algorithms for Minimum Weight Spanning Tree, Counting, Leader Election and related problems, Proceedings of the 9th Annual ACM Symposium on Theory of Computing.

KAYNAKLAR DİZİNİ (devam)

- Baala, H., Flauzac, O., Gaber, J., Bui, M., and El-Ghazawi, T.,** 2003, A self-stabilizing distributed algorithm for spanning tree construction in wireless ad hoc networks, *Journal of Parallel and Distributed Computing*, 63:97-104pp.
- Balazinska, M., Balakrishnan, H., Madden, S. R., and Stonebraker, M.,** 2008, Fault-tolerance in the borealis distributed stream processing system, *ACM Trans. Database Syst.*, 33:1-44pp.
- Beheshti, S. M. R. and Moshkenani, M. S.,** 2007, Development of Grid resource discovery service based on semantic information, pp. 141-148.
- Bell, W., Cameron, D., Millar, P., Capozza, L., Stockinger, K., and Zini, F.,** 2003, Optorsim: A Grid Simulator for Studying Dynamic Data Replication Strategies, *International Journal of High Performance Computing Applications*, 17:403-416pp.
- Bestehorn, M., von der Weth, C., Buchmann, E., and Böhm, K.,** 2010, Fault-tolerant query processing in structured P2P-systems, *Distributed and Parallel Databases*, 28:33-66pp.
- Bizarro, P., Bruno, N., and DeWitt, D. J.,** 2009, Progressive Parametric Query Optimization, *IEEE Trans. on Knowl. and Data Eng.*, 21:582-594pp.
- Blin, L., Potop-Butucaru, M. G., and Rovedakis, S.,** 2009, Self-stabilizing minimum-degree spanning tree within one from the optimal degree, *IPDPS '09: Proceedings of the 2009 IEEE International Symposium on Parallel & Distributed Processing*.
- Bose, S. K., Krishnamoorthy, S., and Ranade, N.,** 2007, Allocating Resources to Parallel Query Plans in Data Grids, *GCC '07: Proceedings of the Sixth International Conference on Grid and Cooperative Computing*.
- Butelle, F., Lavault, C., and Bui, M.,** 1995, A Uniform Self-Stabilizing Minimum Diameter Tree Algorithm (Extended Abstract), *WDAG '95: Proceedings of the 9th International Workshop on Distributed Algorithms*.
- Cai, M., Frank, M., Chen, J., and Szekely, P.,** 2003, MAAN: A multi-attribute addressable network for Grid information services, *4th Int. Workshop on Grid Computing, GRID 2003*.
- Cao, J., Spooner, D., Turner, J. D., Jarvis, S., Kerbyson, D. J., Saini, S., and Nudd, G.,** 2002, Agent-Based Resource Management for Grid Computing, *CCGRID02: Proceedings of the 2nd IEEE/ACM International Symposium on Cluster Computing and the Grid*.

KAYNAKLAR DİZİNİ (devam)

- Chandrasekaran, S., Cooper, O., Deshpande, A., Franklin, M. J., Hellerstein, J. M., Hong, W., Krishnamurthy, S., Madden, S. R., Reiss, F., and Shah, M. A.**, 2003, TelegraphCQ: continuous dataflow processing, Proceedings of the 2003 ACM SIGMOD international conference on Management of data, San Diego, California.
- Cheema, A. S., Muhammad, M., and Gupta, I.**, 2005, Peer-to-Peer Discovery of Computational Resources for Grid Applications, GRID '05: Proceedings of the 6th IEEE/ACM International Workshop on Grid Computing.
- Chihping, W.**, Year, The complexity of processing tree queries in distributed databases, in *Parallel and Distributed Processing, 1990. Proceedings of the Second IEEE Symposium on*, pp. 604-611.
- Cokuslu, D., Hameurlain, A., and Erciyes, K.**, 2009, Grid resource discovery based on web services, International Conference for Internet Technology and Secured Transactions, ICITST 2009.
- Cokuslu, D., Erciyes, K., and Hameurlain, A.**, Year, A Maximum Degree Self-Stabilizing Spanning Tree Algorithm, in *The 25th International Symposium on Computer and Information Sciences (ISCIS 2010)*, pp. 393-396.
- Cokuslu, D., Hameurlain, A., and Erciyes, K.**, 2010b, Grid Resource Discovery Based on Centralized and Hierarchical Architectures, International Journal for Infonomics (IJI), 3:283-292pp.
- Cokuslu, D., Hamuerlain, A., and Erciyes, K.**, 2012a, Resource Allocation for Query Processing in Grid Systems: A Survey, International Journal of Computer Systems Science and Engineering, 27:pp.
- Cokuslu, D., Hamuerlain, A., Erciyes, K., and Morvan, F.**, Year, Resource Allocation Algorithm for a Relational Join Operator in Grid Systems, in *proceedings of 16th International Database Engineering & Applications Symposium, IDEAS12*, Prague, Czech, pp. 139-145.
- Costa, R. L. d. C. and Furtado, P.**, 2008, Scheduling in Grid Databases, Proceedings of the 22nd International Conference on Advanced Information Networking and Applications - Workshops.
- Czajkowski, K., Fitzgerald, S., Foster, I., and Kesselman, C.**, 2001, Grid information services for distributed resource sharing, High Performance Distributed Computing, 2001. Proceedings. 10th IEEE International Symposium on.

KAYNAKLAR DİZİNİ (devam)

- DeWitt, D. J., Katz, R. H., Olken, F., Shapiro, L. D., Stonebraker, M. R., and Wood, D. A.**, 1984, Implementation techniques for main memory database systems, SIGMOD Rec., 14:1-8pp.
- Dijkstra, E. W.**, 1974, Self-stabilizing systems in spite of distributed control, Communications of the ACM, 17:643-644pp.
- Ding, S., Yuan, J., Ju, J., and Hu, L.**, 2005, A Heuristic Algorithm for Agent-Based Grid Resource Discovery, Intl. Conf. on e-Technology, e-Commerce and e-Service.
- Dolev, S.**, 1993, Optimal Time Self Stabilization in Dynamic Systems (Preliminary Version), WDAG '93: Proceedings of the 7th International Workshop on Distributed Algorithms.
- El-Ansary, S.**, 2003, A Framework for the understanding optimization and design of Structured Peer-To-Peer Systems, (unpublished).
- Elmroth, E. and Tordsson, J.**, 2005, An interoperable, standards-based grid resource broker and job submission service, e-Science and Grid Computing, 2005. First International Conference on.
- Epimakhov, I., Hameurlain, A., Dillon, T., and Morvan, F.**, 2011, Resource Scheduling Methods for Query Optimization in Data Grid Systems, Advances in Databases and Information Systems, Springer Berlin / Heidelberg, 185-199.
- Fall, K. and Varadhan, K.**, 2010, The ns Manual (formerly ns Notes and Documentation), (unpublished).
- Fernandez-Baca, D.**, 1989, Allocating modules to processors in a distributed system, Software Engineering, IEEE Transactions on, 15:1427-1436pp.
- Filali, I., Huet, F., and Vergoni, C.**, 2008, A Simple Cache Based Mechanism for Peer to Peer Resource Discovery in Grid Environments, CCGRID '08, Eighth IEEE International Symposium on Cluster Computing and the Grid.
- Fitzgerald, S., Foster, I., Kesselman, C., Laszewski, G. V., Smith, W., and Tuecke, S.**, 1997, A Directory Service for Configuring High-Performance Distributed Computations, In Proc. 6th IEEE Symp. on High Performance Distributed Computing.
- Foster, I. and Kesselman, C.**, 1997, Globus: A metacomputing infrastructure toolkit, International Journal of Supercomputer Applications, 11:115-128pp.
- Foster, I. and Kesselman, C.**, 2004, The Grid: Blueprint for a New Computing Infrastructure, Morgan Kaufmann Publishers, p.

KAYNAKLAR DİZİNİ (devam)

- Fuggetta, A., Picco, G. P., and Vigna, G.**, 1998, Understanding Code Mobility, IEEE Transactions on Software Engineering, 24:342-361pp.
- Gagliardi, F., Jones, B., Reale, M., and Burke, S.**, 2002, European DataGrid Project: Experiences of Deploying a Large Scale Testbed for E-science Applications, Performance Evaluation of Complex Systems: Techniques and Tools, Springer Berlin / Heidelberg, 255-264.
- Gallagher, R. G., Humblet, P. A., and Spira, P. M.**, 1983, A Distributed Algorithm for Minimum-Weight Spanning Trees, ACM Transactions on Programming Languages and Systems, 5:66-77pp.
- Garofalakis, M. N. and Ioannidis, Y. E.**, 1997, Parallel Query Scheduling and Optimization with Time- and Space-Shared Resources, VLDB '97: Proceedings of the 23rd International Conference on Very Large Data Bases, San Francisco, CA, USA.
- Gartner, F. C.**, 2003, A Survey of Self-Stabilizing Spanning-Tree Construction Algorithms, Swiss Federal Institute of Technology (EPFL), School of Computer and Communication Sciences., Technical Report IC/2003/38 June 10, (unpublished).
- Gomoluch, J. and Schroeder, M.**, 2003, Market-Based Resource Allocation for Grid Computing: A Model and Simulation.
- Gounaris, A., Sakellariou, R., Paton, N. W., and Fernandes, A. A. A.**, 2004, Resource Scheduling For Parallel Query Processing On Computational Grids, Grid Computing, 2004. Proceedings. Fifth IEEE/ACM International Workshop on.
- Gounaris, A., Smith, J., Paton, N. W., Sakellariou, R., Fernandez, A. A. A., and Watson, P.**, 2005, Adapting to Changing Resource Performance in Grid Query Processing, DMG, Lecture Notes in Computer Science.
- Gounaris, A., Sakellariou, R., Paton, N. W., and Fernandes, A. A.**, 2006, A novel approach to resource scheduling for parallel query processing on computational grids, Distrib. Parallel Databases, 19:87-106pp.
- Gounaris, A., Smith, J., Paton, N. W., Sakellariou, R., Fernandes, A. A., and Watson, P.**, 2009, Adaptive workload allocation in query processing in autonomous heterogeneous environments, Distrib. Parallel Databases, 25:125-164pp.
- Gupta, S. K. and Srimani, P. K.**, 2003, Self-stabilizing multicast protocols for ad hoc networks, Journal of Parallel and Distributed Computing, 63:87-96pp.

KAYNAKLAR DİZİNİ (devam)

- Hameurlain, A., Morvan, F., and Samad, M. E.**, 2008, Large Scale Data management in Grid Systems: a Survey, IEEE International Conference on Information and Communication Technologies: from Theory to Applications (ICTTA), Damas - Syrie, 07/04/2008-11/04/2008.
- Hameurlain, A., Cokuslu, D., and Erciyes, K.**, 2010, Resource discovery in grid systems: a survey, International Journal of Metadata, Semantics and Ontologies, 5:251-263pp.
- Herault, T., Lemarinier, P., Peres, O., Pilard, L., and Beauquier, J.**, 2006, Self-stabilizing Spanning Tree Algorithm for Large Scale Systems, Stabilization, Safety, and Security of Distributed Systems, 4280:574-575pp.
- Huang, C., Wu, Z., Zheng, G., and Wu, X.**, 2003, Dart: A Framework for Grid-Based Database Resource Access and Discovery, Grid and Cooperative Computing.
- Hwang, J.-H., Balazinska, M., Rasin, A., Cetintemel, U., Stonebraker, M., and Zdonik, S.**, 2005, High-Availability Algorithms for Distributed Stream Processing, Proceedings of the 21st International Conference on Data Engineering.
- Iamnitchi, A. and Foster, I.**, 2003, Chapter 1 A Peer-to-Peer Approach to Resource Location in Grid Environments, Kluwer.
- Iamnitchi, A. and Talia, D.**, 2005, P2P computing and interaction with grids, Future Generation Computer Systems, 21:331-332pp.
- Ioannidis, Y. E., Ng, R. T., Shim, K., and Sellis, T. K.**, 1997, Parametric query optimization, The VLDB Journal, 6:132-151pp.
- Jeong-hyon Hwang, Ying Xing, and Zdonik, S.**, Year, A cooperative, self-configuring high-availability solution for stream processing, in *ICDE*.
- Jian, Y. and Liu, Y.**, 2007, The State of the Art in Grid Scheduling Systems, Proceedings of the Third International Conference on Natural Computation - Volume 04.
- Jiang, C., Wang, C., Liu, X., and Zhao, Y.**, 2007, A survey of job scheduling in grids, Proceedings of the joint 9th Asia-Pacific web and 8th international conference on web-age information management conference on Advances in data and web management, Huang Shan, China.
- Jun, K., Boloni, L., Palacz, K., and Marinescu, D. C.**, 2000, Agent-Based Resource Discovery, in Proc. 9th IEEE Heterogeneous Computing Workshop.

KAYNAKLAR DİZİNİ (devam)

- Kakarontzas, G. and Savvas, I. K.**, 2006, Agent-Based Resource Discovery and Selection for Dynamic Grids, Proc of the 15th IEEE Intl. Workshops on Enabling Technologies.
- Kant, U. and Grosu, D.**, 2005, Double Auction Protocols for Resource Allocation in Grids, ITCC '05: Proceedings of the International Conference on Information Technology: Coding and Computing (ITCC'05) - Volume I.
- Karaata, M. H., Pemmaraju, S. V., Bruell, S. C., and Ghosh, S.**, 1994, Self-stabilizing algorithms for finding centers and medians of trees, PODC94, Proceedings of the thirteenth annual ACM symposium on Principles of distributed computing, New York, NY, USA.
- Kaur, D. and Sengupta, J.**, 2007, Resource Discovery in Web-Services Based Grids, Proceedings Of World Academy Of Science, Engineering And Technology.
- Kosowski, A. and Kuszner, L.**, 2006, A Self-stabilizing Algorithm for Finding a Spanning Tree in a Polynomial Number of Moves, Paralel and Distributed Non-numerical Algorithms, 3911/2006:75-82pp.
- Kotowski, N., Lima, A. A. B., Pacitti, E., Valduriez, P., and Mattoso, M.**, 2008, Parallel query processing for OLAP in grids, Concurr. Comput. : Pract. Exper., 20:2039-2048pp.
- Krauter, K., Buyya, R., and Maheswaran, M.**, 2002, A taxonomy and survey of grid resource management systems for distributed computing, Softw. Pract. Exper., 32:135-164pp.
- Kshemkalyani, A. D. and Singhal, M.**, 2008, Distributed Computing: Principles, Algorithms, and Systems, Cambridge University Press, p.
- Kwon, Y., Balazinska, M., and Greenberg, A.**, 2008, Fault-tolerant stream processing using a distributed, replicated file system, Proc. VLDB Endow., 1:574-585pp.
- Li, W., Xu, Z., Dong, F., and Zhang, J.**, 2002, Grid Resource Discovery Based on a Routing-Transferring Model, pp. 145-156.
- Lien, Y. N.**, 1988, A New Node-Join-Tree Distributed Algorithm for Minimum Weight Spanning Trees, Proceedings of the 8th International Conference on Distributed Computing System.
- Liu, S. and Karimi, H. A.**, 2008, Grid query optimizer to improve query processing in grids, Future Gener. Comput. Syst., 24:342-353pp.

KAYNAKLAR DİZİNİ (devam)

- Mandal, A., Kennedy, K., Koelbel, C., Marin, G., Mellor-Crummey, J., Liu, B., and Johnsson, L.**, 2005, Scheduling Strategies For Mapping Application Workflows Onto The Grid, Proceedings, 14th IEEE International Symposium on High Performance Distributed Computing.
- Manvi, S. S., Birje, M. N., and Prasad, B.**, 2005, An Agent-based Resource Allocation Model for Grid Computing, IEEE SCC.
- Marzolla, M., Mordacchini, M., and Orlando, S.**, 2005, Resource discovery in a dynamic Grid environment, DEXA Workshop 2005.
- Marzolla, M., Mordacchini, M., and Orlando, S.**, 2007, Peer-to-peer systems for discovering resources in a dynamic grid, Parallel Comput., 33:339-358pp.
- Mastroianni, C., Talia, D., and Verta, O.**, 2005, A super-peer model for resource discovery services in large-scale Grids, Future Generation Computer Systems.
- Mattoso, M., Zimbrão, G., Lima, R. A. B., Baião, A., Braganholo, V. P., Aveleda, A., Mir, B., Almentero, B. K., and Costa, M. N.**, 2005, ParGRES: a middleware for executing OLAP queries in parallel, COPPE/UFRJ, Technical Report ES-690, (unpublished).
- Molto, G., Hernandez, V., and Alonso, J. M.**, 2008, A service-oriented WSRF-based architecture for metascheduling on computational Grids, Future Generation Computing Systems, 24:317-328pp.
- Oppenheimer, D., Albrecht, J., Patterson, D., and Vahdat, A.**, 2004, Scalable Wide-Area Resource Discovery, Technical Report CSD04 -1334, University of California Berkeley.
- Özsu, M. T. and Valduriez, P.**, 2011, Principles of Distributed Database Systems, Third Edition, Springer, 120p.
- Pacitti, E., Valduriez, P., and Mattoso, M.**, 2007, Grid Data Management: Open Problems and New Issues, Journal of Grid Computing, 5:273-281pp.
- Pan, R. C., Wang, J. Z., and Chow, L. R.**, 1999, A self-stabilizing distributed spanning tree construction algorithm with a distributed demon, Tamsui Oxford Journal of Mathematical Sciences, 15:23-32pp.
- Paton, N. W., Buenabad-Chavez, J., Chen, M., Raman, V., Swart, G., Narang, I., Yellin, D. M., and Fernandes, A. A.**, 2009, Autonomic query parallelization using non-dedicated computers: an evaluation of adaptivity options, The VLDB Journal, 18:119-140pp.

KAYNAKLAR DİZİNİ (devam)

- Pourebrahimi, B. and Bertels, K.**, 2007, Fair access to scarce resources in ad-hoc grids using an economic-based approach, Proceedings of the 5th international workshop on Middleware for grid computing: held at the ACM/IFIP/USENIX 8th International Middleware Conference, Newport Beach, California.
- Puh, M., Jezic, G., and Kusek, M.**, 2007, Multi-agent System for Resource Discovery in Grid Network, 16th IEEE International Workshops on Enabling Technologies: Infrastructures for Collaborative Enterprises.
- Puppini, D., Moncelli, S., Baraglia, R., Tonelotto, N., and Silvestri, F.**, 2005, A Grid information service based on Peer-to-Peer, EuroPar, Springer LNCS.
- Quiane-Ruiz, J.-A., Lamarre, P., and Valduriez, P.**, 2009, A self-adaptable query allocation framework for distributed information systems, The VLDB Journal, 18:649-674pp.
- Quiane-Ruiz, J.-A., Lamarre, P., and Valduriez, P.**, 2009, A self-adaptable query allocation framework for distributed information systems, The VLDB Journal, 18:649-674pp.
- Ramos, T. G. and Magalhaes, A. C.**, 2006, An Extensible Resource Discovery Mechanism for Grid Computing Environments, CCGRID '06: Proceedings of the Sixth IEEE International Symposium on Cluster Computing and the Grid.
- Riedel, M., Schuller, B., Mallmann, D., Menday, R., Streit, A., Tweddell, B., Memon, M. S., Memon, A. S., Demuth, B., and Lippert, T.**, 2007, Web Services Interfaces and Open Standards Integration into the European UNICORE 6 Grid Middleware, EDOCW '07: Proceedings of the 2007 Eleventh International IEEE EDOC Conference Workshop.
- Rowstron, A. and Druschel, P.**, 2001, Pastry: Scalable, decentralized object location, and routing for large-scale peer-to-peer systems, Lecture Notes in Computer Science.
- Silva, V. F. V. D., Dutra, M. L., Porto, F., Schulze, B., Barbosa, A. C., and de Oliveira, J. C.**, 2006, An adaptive parallel query processing middleware for the Grid: Research Articles, Concurr. Comput. : Pract. Exper., 18:621-634pp.
- Slimani, Y., Najjar, F., and Mami, N.**, 2004, An Adaptive Cost Model for Distributed Query Optimization on the Grid, OTM Workshops.

KAYNAKLAR DİZİNİ (devam)

- Smith, J. and Watson, P.**, Year, Fault-tolerance in distributed query processing, in *Database Engineering and Application Symposium, 2005. IDEAS 2005. 9th International*, pp. 329-338.
- Smith, J. and Watson, P.**, 2007, Failure Recovery Alternatives in Grid-Based Distributed Query Processing: A Case Study, *Knowledge and Data Management in GRIDs*, Springer US, 51-63.
- Soe, K. M., Nwe, A. A., Aung, T. N., Naing, T. T., and Thein, N. L.**, 2005, Efficient Scheduling of Resources for Parallel Query Processing on Grid-based Architecture, *Information and Telecommunication Technologies, 2005. APSITT 2005 Proceedings. 6th Asia-Pacific Symposium on*.
- Spence, D. and Harris, T.**, 2003, Xenosearch: Distributed resource discovery in the xenoserver open platform, In *Proceedings of HPDC*.
- Sulistio, A., Cibej, U., Venugopal, S., Robic, B., and Buyya, R.**, 2008, A toolkit for modelling and simulating data Grids: an extension to GridSim, *Concurr. Comput. : Pract. Exper.*, 20:1591-1609pp.
- Talia, D. and Trunfio, P.**, 2005, Peer-to-Peer protocols and Grid services for resource discovery on Grids, *Advances in Parallel Computing*, 14:83-103pp.
- Talia, D., Trunfio, P., and Zeng, J.**, 2007, Peer-to-Peer Models for Resource Discovery in Large-Scale Grids: A Scalable Architecture, *High Performance Computing for Computational Science - VECPAR 2006*.
- Tang, X. and Huang, L.**, 2006, Grid Resource Management Based on Mobile Agent, *WISE Workshops*.
- Taniar, D., Leung, C. H. C., Rahayu, W., and Goel, S.**, 2008, *High Performance Parallel Database Processing and Grid Databases*, Wiley Publishing, p.
- Taylor, N. E.**, 2008, Recovery from Node Failure in Distributed Query Processing, *University of Pennsylvania Department of Computer and Information Science, MS-CIS-08-38*, (unpublished).
- Trunfio, P., Talia, D., Papadakis, H., Fragopoulou, P., Mordacchini, M., Pennanen, M., Popov, K., Vlassov, V., and Haridi, S.**, 2007, Peer-to-Peer resource discovery in Grids: Models and systems, *Future Gener. Comput. Syst.*, 23:864-878pp.
- Venugopal, S., Buyya, R., and Winton, L.**, 2006, A Grid service broker for scheduling e-Science applications on global data Grids: Research Articles, *Concurr. Comput. : Pract. Exper.*, 18:685-699pp.

KAYNAKLAR DİZİNİ (devam)

- Yan, M., Yu-hui, Q., Gang, W., and Ju-Hua, Z.**, 2007, Study of Grid Resource Discovery Based on Mobile Agent, Proc. of the 3rd Intl. Conf. on Semantics, Knowledge and Grid.
- Yang, B. and Garcia-Molina, H.**, 2003, Designing a super-peer network, Data Engineering, 2003. Proceedings. 19th International Conference on.
- Yin, Y., Cui, H., and Chen, X.**, 2007, The Grid Resource Discovery Method Based on Hierarchical Model, Information Technology Journal, 6:1090-1094pp.
- Yu, J., Venugopal, S., and Buyya, R.**, 2003, Grid Market Directory: A Web Services based Grid Service Publication Directory, Grid Computing and Distributed Systems (GRIDS) Lab, Dept. of Computer Science and Software Engineering, The University of Melbourne, (unpublished).
- Yu, J., Zhao, C., and Pan, Y.**, 2006, Grid Resource Management Based on Mobile Agent, Proc. of the Intl. Conf. on Computational Intelligence for Modeling Control and Automation and Intl. Conf. on Intelligent Agents Web Technologies and Intl Commerce.
- Zhao, X., Xu, L., and Wang, B.**, 2007, A Resource Allocation Model with Cost-Performance Ratio in Data Grid, SNPD '07: Proceedings of the Eighth ACIS International Conference on Software Engineering, Artificial Intelligence, Networking, and Parallel/Distributed Computing.

ÖZGEÇMİŞ

Deniz ÇOKUSLU

Adres: İzmir Yüksek Teknoloji Enstitüsü 35430 Urla/İZMİR

e-mail: denizcokuslu@iyte.edu.tr

Eğitim Durumu

- Doktora : 2007 - 2012, Ege Üniversitesi, Uluslararası Bilgisayar Enstitüsü, İzmir, Türkiye
2008 - 2012, Paul Sabatier Üniversitesi, Toulouse, Fransa
- Yüksek Lisans : 2004 – 2007, İzmir Yüksek Teknoloji Enstitüsü, Bilgisayar Mühendisliği Bölümü, İzmir, Türkiye
- Lisans : 1999 – 2004, İzmir Yüksek Teknoloji Enstitüsü, Bilgisayar Mühendisliği Bölümü, İzmir, Türkiye

İş Deneyimi

- Öğretim Görevlisi : 2012 - , İzmir Yüksek Teknoloji Enstitüsü, Bilgisayar Mühendisliği Bölümü, İzmir, Türkiye
- Araştırma Görevlisi : 2004 – 2012, İzmir Yüksek Teknoloji Enstitüsü, Bilgisayar Mühendisliği Bölümü, İzmir, Türkiye
- Yazılım Uzmanı : 2003 – 2004, Yaşar Holding, Astron Proje Ofisi, İzmir, Türkiye

Yabancı Dil

- Türkçe : Anadil
- Fransızca : İleri seviye
- İngilizce : İleri seviye

Bilgisayar dilleri ve program deneyimi

- C/C++, Java, Abap, TCL
- NS2, TOSSİM Benzetim ortamları

Yayınlar

Bildiriler

- **Cokuslu, D., Erciyes, K., and Dagdeviren, O.**, 2006, A Dominating Set Based Clustering Algorithm for Mobile Ad Hoc Networks, Computational Science, ICCS 2006, Springer LNCS.
- **Cokuslu, D. and Erciyes, K.**, 2007, A Hierarchical Connected Dominating Set Based Clustering Algorithm for Mobile Ad Hoc Networks, 15th International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems, MASCOTS '07, IEEE.
- **Cokuslu, D. and Erciyes, K.**, 2008, A flooding based routing algorithm for Mobile Ad Hoc Networks, 16th Signal Processing, Communication and Applications Conference, SIU2008, IEEE.
- **Cokuslu, D., Hameurlain, A., and Erciyes, K.**, 2009, Grid resource discovery based on web services, International Conference for Internet Technology and Secured Transactions, ICITST 2009.
- **Cokuslu, D., Erciyes, K., and Hameurlain, A.**, Year, A Maximum Degree Self-Stabilizing Spanning Tree Algorithm, in *The 25th International Symposium on Computer and Information Sciences (ISCIS 2010)*, pp. 393-396.
- **Cokuslu, D., Hamuerlain, A., Erciyes, K., and Morvan, F.**, Year, Resource Allocation Algorithm for a Relational Join Operator in Grid Systems, in *proceedings of 16th International Database Engineering & Applications Symposium, IDEAS12*, Prague, Czech, pp. 139-145.
- **Dagdeviren, O., Erciyes, K., and Cokuslu, D.**, 2005, Merging Clustering Algorithms in Mobile Ad hoc Networks, Distributed Computing and Internet Technology, ICDCIT05, Springer LNCS.
- **Dagdeviren, O., Erciyes, K., and Cokuslu, D.**, 2006, A merging clustering algorithm for mobile ad hoc networks, In Proceedings of International Conference on Computational Science and Its Applications 2006, ICCSA06.

Makaleler

- **Cokuslu, D., Hameurlain, A., and Erciyes, K.,** 2010, Grid Resource Discovery Based on Centralized and Hierarchical Architectures, International Journal for Infonomics (IJI), 3:283-292pp.
- **Cokuslu, D., Hamuerlain, A., and Erciyes, K.,** 2012, Resource Allocation for Query Processing in Grid Systems: A Survey, International Journal of Computer Systems Science and Engineering, 27:pp.
- **Dagdeviren, O., Erciyes, K., and Cokuslu, D.,** 2012, Modified Merging Clustering and Backbone Formation Algorithms for Mobile Ad Hoc Networks, Applied and Computational Mathematics, 11:27pp.
- **Erciyes, K., Dagdeviren, O., Cokuslu, D., and Ozsoyeller, D.,** 2007, Graph Theoretic Clustering Algorithms in Mobile Ad Hoc Networks and Wireless Sensor Networks, Applied and Computational Mathematics, 6/2:162-180pp.
- **Hameurlain, A., Cokuslu, D., and Erciyes, K.,** 2010, Resource discovery in grid systems: a survey, International Journal of Metadata, Semantics and Ontologies, 5:251-263pp.

Kitaplar

- **Cokuslu, D., Erciyes, K., and Dagdeviren, O.,** 2010, Topology Control for Mobile Ad hoc Networks: A Dominating Set Based Approach, VDM Verlag, p.
- **Dagdeviren, O., Erciyes, K., and Cokuslu, D.,** 2010, A Communication Architecture for Mobile Ad hoc Networks: Cluster based Protocols for Distributed Applications, VDM Verlag, p.

Kitap Bölümleri

- **Erciyes, K., Dagdeviren, O., Cokuslu, D., Yılmaz, O., and Gumus, H.,** 2011, Modeling and Simulation Tools for Mobile Ad hoc Networks, CRC Press,

Tezler

- **Cokuslu, D.,** 2007, A Dominating Set Based Communication Architecture for Distributed Applications in Mobile Ad Hoc Networks, Izmir Institute of Technology.