EGE ÜNİVERSİTESİ FEN BİLİMLERİ ENSTİTÜSÜ

(DOKTORA TEZİ)

DAĞITIK SORGU İŞLEMEDE KAYNAK ATAMA İÇİN ÇİZGE MERKEZİLİK ALGORİTMALARI

Vedat KAVALCI

Tez Danışmanı: Doç. Dr. Orhan DAĞDEVİREN

Uluslararası Bilgisayar Anabilim Dalı

Sunuş Tarihi: 23.07.2018

BORNOVA-İZMİR
2018

Vedat KAVALCI tarafından Doktora tezi olarak sunulan "*Dağıtık Sorgu İşlemede Kaynak Atama İçin Çizge Merkezilik Algoritmaları*" başlıklı bu çalışma EÜ Lisansüstü Eğitim ve Öğretim Yönetmeliği ile EÜ Fen Bilimleri Enstitüsü Eğitim ve Öğretim Yönergesinin ilgili hükümleri uyarınca tarafımızdan değerlendirilerek savunmaya değer bulunmuş ve 23.07.2018 tarihinde yapılan tez savunma sınavında aday oybirliği/oyçokluğu ile başarılı bulunmuştur.

Jüri Üyeleri:		**İmza**
Jüri Başkanı	**: Doç. Dr. Orhan DAĞDEVİREN**	
Raportör Üye	**: Doç. Dr. Müge Sayıt**	
Üye	**: Prof. Dr. Alp Kut**	
Üye	**: Prof. Dr. Aylin Kantarcı**	
Üye	**: Dr. Öğr. Üyesi Kaya Oğuz**	

EGE ÜNİVERSİTESİ FEN BİLİMLERİ ENSTİTÜSÜ

ETİK KURALLARA UYGUNLUK BEYANI

EÜ Lisansüstü Eğitim ve Öğretim Yönetmeliğinin ilgili hükümleri uyarınca Yüksek Lisans Tezi / Doktora Tezi olarak sunduğum "*Dağıtık Sorgu İşlemede Kaynak Atama İçin Çizge Merkezilik Algoritmaları*" başlıklı bu tezin kendi çalışmam olduğunu, sunduğum tüm sonuç, doküman, bilgi ve belgeleri bizzat ve bu tez çalışması kapsamında elde ettiğimi, bu tez çalışmasıyla elde edilmeyen bütün bilgi ve yorumlara atıf yaptığımı ve bunları kaynaklar listesinde usulüne uygun olarak verdiğimi, tez çalışması ve yazımı sırasında patent ve telif haklarını ihlal edici bir davranışımın olmadığını, bu tezin herhangi bir bölümünü bu üniversite veya diğer bir üniversitede başka bir tez çalışması içinde sunmadığımı, bu tezin planlanmasından yazımına kadar bütün safhalarda bilimsel etik kurallarına uygun olarak davrandığımı ve aksinin ortaya çıkması durumunda her türlü yasal sonucu kabul edeceğimi beyan ederim.

23 / 07 / 2018

İmzası

Vedat KAVALCI

ÖZET

Dağıtık Sorgu İşlemede Kaynak Atama İçin Çizge Merkezilik Algoritmaları

KAVALCI, Vedat

Doktora Tezi, Uluslararası Bilgisayar Anabilim Dalı
Tez Danışmanı: Doç. Dr. Orhan DAĞDEVİREN
Ağustos 2018, 104 sayfa

Bu çalışmada merkezilik tabanlı algoritmaların ve dolayısı ile düğümlere ait ilingesel özelliklerinin sorgu işleme sisteminin performansı üzerine olan etkileri teorik ve pratik olarak incelenmiştir. Öncelikle, bulunabilen dağıtık mimarideki merkezilik algoritmaları analiz edilmiş, sınıflandırılmış ve karmaşık ağlar için uygunlukları araştırılmıştır. Daha sonra, dağıtık sorgu işleme sisteminin ihtiyaçları dikkate alınarak tasarlanan üç adet merkezilik tabanlı kaynak atama algoritması önerilmiştir. Bu algoritmalar, daha doğru adaylarının daha kısa sürede seçilmesini amaçlayan sezgisel yaklaşımlar içermekte ve her biri kendi merkezilik sınıfının en temel özelliklerini taşımaktadır. Son olarak, önerilen algoritmalar ile kodlanan kıyas algoritmalarının teorik ve deneysel analizleri yapılmıştır.

Gerçek hayat şartları dikkate alınarak yapılan benzetimler sonucunda, önerilen CCBC ile BCBC algoritmaları birbirlerine çok yakın ve en iyi sonuçları vermiştir. Bu durum, yakınlık ve aradalılık merkezilik yaklaşımlarının, kaynak atama sürecinde olumlu ve benzer oranda etkili olduğunu göstermiştir. Buradan yola çıkarak, dağıtık sorgu işleme sistemlerinde, düğümlerin ilingesel özelliklerin oldukça etkili olduğu ve kaynak atama aşamasında kullanılan maliyet modelinde bir parametre olarak kullanılmasının uygun olduğu sonucuna varılmıştır. Ayrıca bu çalışmada önerilen; sınırlı merkezilik yaklaşımı ve alt çizge belirleme yaklaşımlarının aday belirleme sürecinde algoritma maliyetlerini olumlu yönde etkilediği gözlenmiştir. Dağıtık sorgu işleme sistemi ile merkezilik hesaplama yöntemleri dikkate alınarak tasarlanan bu yaklaşımlar sadece aday kümesini daraltmakla kalmayıp doğru adayın seçimine de katkı sağlamışlardır.

Anahtar sözcükler: Karmaşık ağlar, merkezilik, dağıtık programlama, sorgu işleme, kaynak atama.

ABSTRACT

Graph Centrality Algorithms For Resource Allocation In Distributed Query Processing

KAVALCI, Vedat

PhD in International Computer Department.
Supervisor: Assoc. Prof. Dr. Orhan DAĞDEVİREN
August 2018, 104 pages

In this study, the effectiveness of the topological properties of the nodes has been investigated with help of the centrality based algorithms. Firstly, recently distributed centrality algorithms have been analyzed, clustered and investigated their applicability of algorithms to complex networks. Then, three centrality based resource allocation algorithms have been proposed. These algorithms are designed taking into account the needs of the distributed query processing. Proposed algorithms have some heuristic approaches to be fast and to find correct candidates. Besides, proposed algorithms are good example of their class. Finally, theoretical and experimental analyzes of the compare and proposed algorithms have been carried out.

Real-life conditions are taken into consideration in simulation studies. As a result of the simulation studies, the CCBC and BCBC algorithms are very close to each other and gave the best results. This shows that closeness and betweenness centrality approaches have a positive and similar effect on the resource allocation process. Based on this; in distributed query processing systems, it has been concluded that the topological properties of the nodes are very effective and it is appropriate to use them as a parameter in the cost model used in the resource allocation phase. Besides, in this study, proposed limited centrality approach and subgraph detection approaches have been observed to affect the algorithm costs positively in the candidate determination process. These approaches, which are designed by considering the distributed query processing system and centrality calculation methods, not only narrow the candidate cluster but also contribute to the selection of the correct candidate.

Keywords: Complex networks, centlality, distributed programing, query processing, resource allocation.

TEŞEKKÜR

Bu çalışmayı değerli bilgileri ve deneyimleriyle yönlendiren danışman hocalarım sayın Dr. Deniz ÇOKUSLU ve sayın Doç. Dr. Orhan DAĞDEVİREN'e çok teşekkür ederim. Değerli önerileri ve çalışmaya olan katkılarından dolayı hocam sayın Prof. Dr. Kayhan ERCİYES'e ayrıca teşekkür ederim.

Bu uzun ve yoğun çalışma sürecinde, yaşanan tüm olumsuzluklara rağmen benden desteğini hiçbir zaman eksik etmeyen aile fertlerime ve özellikle eşime ayrıca teşekkür ederim.

İÇİNDEKİLER

Sayfa

İÇİNDEKİLER (devam)

Sayfa

İÇİNDEKİLER (devam)

Sayfa

ŞEKİLLER DİZİNİ

Şekil Sayfa

ŞEKİLLER DİZİNİ (devam)

Şekil Sayfa

ÇİZELGELER DİZİNİ

Çizelge Sayfa

1. GİRİŞ

Günümüzde, gelişen bilgi işlem teknolojileri ve kullanılmakta olan veri boyutunun büyüklüğü nedeniyle sorgu işleme (İng. query processing) sistemleri oldukça yaygın hale gelmiştir. Dağıtık sorgulama sistemleri ise alternatifi olan merkezi sorgu işleme sistemlerinin yetersizliklerini gidermek için geliştirilmiştir. Bu yaklaşımla, alternatiflerine göre hem daha esnek hem de ölçeklenebilir (İng. scalable) çözümler üretilebilmektedir (Ozsu 2007). Ayrıca bilgiye olan ihtiyacın her geçen gün daha da fazla artması ve kullanılmakta olan veri miktarının hızlı büyümesinden dolayı özellikle dağıtık sorgulama sistemlerinin gelecekte daha da yaygın kullanım alanı bulması beklenmektedir. Bu durum, pek çok araştırmacıyı bu alanda çalışmaya sevk etmiştir (Stonebraker et al. 1996; Kotowski et al. 2008).

Dağıtık sorgulama sistemleri; ağların büyük boyutlu olması (İng. large scale), (çok sayıda kullanıcı içermesi, sistemlerin farklı ve uzak yerlerde olması, karmaşık ağ yapılarına sahip oluşması vb.), özdeş olmayan sistemlerden oluşması, devingen bir yapıya sahip olması, kullanıcıların hızlı ve güvenilir sistemlere olan ihtiyacı gibi özelliklerden dolayı bilgi işlem teknolojilerinin sınırlarını zorlamaktadır (Çokuslu et al. 2012a; Erciyes 2014). Bu durum dağıtık sorgu işleme sistemlerinde sürekli yeni problemlerin doğmasına neden olmaktadır. Bu tür problemlerden birisi de giderek artmakta olan ağ trafiğidir. Çalışmamızın çıkış noktasını oluşturan bu problem, dağıtık sorgu işleme sistemlerindeki önemli problemlerden biridir ve veri kaynaklarının (İng. data resource) sayısına, gerçekleştirilecek sorgu türüne ve veri miktarına bağlı olarak yüksek boyutlara ulaşabilmektedir (Dedzoe et al. 2010). Sorgu işleme sistemlerinde genellikle veri akış yönü, veri kaynaklarından sorgu işleme birimlerine doğrudur. Sorgu işleme birimlerinin ve veri kaynaklarının ağda birbirlerine göre olan konumları, oluşacak ağ trafiği üzerinde etkilidir. Kullanılacak veri kaynakları sorgulamanın içeriğine göre belirlenmekte ve aynı veriyi içeren başka bir veri kaynağı yoksa alternatif bir veri kaynağı da olamamaktadır. Oysa sorgu işleme birimlerinin/kaynakların seçiminde ilingesel (İng. topological) ve donanımsal özellikler dikkate alınarak farklı alternatifler oluşturulabilmektedir. Bu durum, sistemde sorgu işleminde kullanılacak birimlerin seçimini, yani kaynak atama (İng. resource allocation) işleminin önemini arttırmaktadır. Genel olarak kaynak atama, sorgu süresini en aza indirecek şekilde sorgu operatörlerinin işletilmesi için uygun düğümlerin seçilmesi ve ayrılması olarak tanımlanabilir (Çokuslu 2012c). Bu problem, büyük ölçekli sorgu işleme sistemleri için polinom zamanda çözümü olmadığı (İng. NP-

Comlete) ispat edilmiş bir eniyileme (İng. optimization) problemidir (Chihping 1990; Ozsu 2007).

Sorgu işleme sistemlerinde en yoğun veri trafiğine maruz kalan birimlerden birisi de veri birleştirme operatörleridir (İng. join operator). Bu nedenle bu tez çalışmasında veri birleştirme operatörlerinin seçimi problemine odaklanılmış ve farklı bir çözüm üretebilmek amacıyla yeni kaynak atama algoritmaları önerilmiştir.

Önerilen algoritmalar çizge merkezilik (İng. graph centrality) yaklaşımına dayandırılmıştır. Çizge merkezilik kavramı yeni bir yaklaşım değildir, ama son yılların yaygın kullanılan araştırma alanlarından birisidir. Çizge merkezilik kavramı, belirli bir çizgedeki göreceli olarak önemli olan nesneleri (düğüm/kenar İng. node/edge) bulmak için kullanılmaktadır. Dağıtık sorgu işleme sistemlerinde, merkezilik tabanlı kaynak atama algoritmalarının tasarlanması ise yeni bir yaklaşımdır. Bildiğimiz kadarı ile bu alanda böyle bir çalışmaya rastlanmamıştır (Çokuslu et al. 2012a; Çokuslu et al. 2012b). Bu çalışmadaki en önemli katkılar, dağıtık sorgu işlemlerinde iletişim maliyetlerini düşürmek, sorgu sürecini kısaltmak ve ölçeklenebilir (İng. scalable) çözümler üretmek olacaktır.

Bu çalışmanın hedefi, ağ trafiğini en aza indirgeyebilmek amacıyla, dağıtık sorgu işleme sistemleri için merkezilik tabanlı kaynak atama algoritmaları önermektir. Bu nedenle üç yeni algoritma, iki tane de kıyas algoritması kodlanmıştır. Algoritmaların başarısı, veri kaynakları ve veri birleştirme operatörleri arasındaki iletişim süreleri kıyaslanarak test edilmiştir. Önerilen algoritmalar, veri kaynakları arasındaki veri iletişim maliyetini en aza indirgeyebilecek aday düğümlerin (İng. node) listesini oluşturmaktadır ve bu liste, ağdaki veri kaynağı olmayan düğümleri de içerebilmektedir. Bu amaç için öncelikle, bir alt çizge (İng. subgraph) belirleme algoritması önerilmiştir. Esas çizgeye oranla daha az düğüm içermesi beklenen alt çizge, tüm veri kaynaklarıyla birlikte bunların arasında kalan bazı sıradan düğümleri içermektedir. Önerilen algoritmalar alt çizge belirleme algoritmasından sonra çalıştırılmakta ve merkezilik tipine göre etkin aday listeleri elde edilmektedir. Kıyaslama yapabilmek amacıyla, daha önceki araştırmalarda önerilmiş olan aday belirleme algoritmaları da aynı çizgeler üzerinde çalıştırılmış ve onların da aday listesi oluşturması sağlanmıştır. Daha sonra, kıyas algoritmalarının belirlediği adaylar ile önerilen algoritmaların adayları sorgu işleme benzetimi yardımıyla test edilmiş ve algoritmaların başarısı karşılaştırılmıştır.

1.1 Dağıtık Sorgu İşlemede Kaynak Atama

Dağıtık sorgu işleme süreci, sorgunun alınması ve çözümlenmesi (İng. parsing) ile başlayıp sonucun elde edilmesine kadar devam eden birçok aşamadan oluşmaktadır (Kossmann 2000). Kaynak atama ise sorgu eniyileme (İng. query optimization) amacıyla gerçekleştirilen adımlardan biridir. Genel olarak; sorgulama sürecinin verimini artırmak amacıyla uygun kaynağın, daha önce tanımlanmış olan maliyet modelindeki (İng. cost model) kısıtlara göre seçilmesi olarak tanımlanmaktadır (Kossmann 2000). Büyük ve karmaşık ağlarda, en iyi kaynak atama işleminin polinom zamanda çözümünün olmadığı ispat edildiği için (Chihping 1990) problemin çözümünde genellikle sezgisel yaklaşımlar kullanılmaktadır. Ayrıca bildiğimiz kadarı ile araştırma aşamasında Çokuslu'nun çalışması(Çokuslu 2012c) dışında, kaynak atama işlemlerinde sınırlı çizge verisi kullanımına yönelik bir çalışmaya rastlanmamıştır. Bu özellik, kaynak atama algoritmalarına ölçeklenebilirlik ve hız kazandırması açısından oldukça önemlidir. Kaynak atama sayesinde; *i.* kaç tane kaynak kullanılacağı, *ii.* sorgu işleme anında hangi kaynakların ilave olarak kullanılabileceği ve *iii.* sorgu işleme anında, kaynak düğümlerden birinde bir sorun çıktığında, yerine kullanılabilecek kaynak düğüm kolaylıkla belirlenebilmektedir (Çokuslu 2012c).

Bu tez çalışmasında, kaynak atama sürecindeki veri birleştirme operatörlerinin seçimine odaklanılmıştır. Veri birleştirme operatörlerinin seçimi sorgu işlemeden önceki son aşamalardan biridir. Sistem, kaynak keşfi (İng. resource discovery) aşamasını tamamladıktan sonra veri birleştirme işlemini yapabilecek düğümler arasından uygun adayları maliyet modeline göre belirlemektedir. Adaylar, veri kaynağı olup olmamasına bakılmaksızın çizgedeki düğümler arasından seçilmektedir. Daha sonra, sorgu işleminin verimini eniyileyecek şekilde hazırlanan kaynak listesindeki düğümlerin uygunluk dereceleri dikkate alınarak, bir ya da birden fazlası kullanılarak sorgu süreci tamamlanmaktadır.

1.2 Problem ve Motivasyon

Kaynak atamadaki amaç, sorgulama işlemini en kısa sürede tamamlayacak ve en az veri trafiği oluşturacak adayların seçilmesidir. Bir başka deyişle, bu bir eniyileme (İng. optimization) problemidir. Polinom zamanda (İng. polynomial time) bir çözümünün olamayan bu problemin NP-Complete bir problem olduğu ispatlanmıştır (Chihping 1990; Ozsu 2007). Bu yüzden problemin çözümünde

önerilen algoritmalar genellikle sezgisel (İng. heuristic) ya da açgözlü (İng. greedy) yaklaşımlar içermektedir (Gounaris et al. 2004; Silva et al. 2006; Gounaris et al. 2006a; Gounaris et al. 2006b; Kotowski et al. 2008; Liu and Karimi 2008; Gounaris et al. 2009; Bose et al. 2013).

Kaynak atama konusunda var olan çalışmaların çoğunda maliyet modeli olarak adlandırılan ve seçim kıstaslarını içeren bir listeden faydalanılarak adaylar belirlenmektedir. Bu kıstaslar, ilgili düğüme ait işlemci gücü, disk kapasitesi, giriş-çıkış hızı, bellek kapasitesi, bulundurduğu tablolar, yüklü programlar ve iletişim hızı gibi özellikleri içermektedir. Kullanılan maliyet modelleri incelendiğinde, genellikle ilgili düğümde bulunan veriye ait kıstaslar ve ilgili düğümün donanımsal özelliklerine yönelik kıstasların dikkate alındığı tespit edilmiştir (Gounaris et al. 2006a; Liu and Karimi 2008; Bose et al. 2013). İletişim maliyeti olarak ise sadece ilgili düğüme ait kanal genişliğinin (İng. bandwidth) hesaba katıldığı gözlenmiştir (Gounaris et al. 2006a; Liu and Karimi 2008; Bose et al. 2013). Bildiğimiz kadarı ile Çokuslu ve arkadaşlarının çalışması hariç (Çokuslu et al. 2012b) daha önceki çalışmalarda öne sürülen maliyet modelinde, ilgili düğümün ağdaki konumuna yönelik ilingesel bir kıstasa rastlanmamıştır. Ayrıca dağıtık sorgu sistemleri genellikle örgü (İng. grid) veya internet gibi büyük bilgisayar ağlarının bir parçası olmasına rağmen yani çok sayıda düğüm içeren karmaşık ağlarda bulunmasına rağmen, bildiğimiz kadarı ile yine Çokuslu ve arkadaşları hariç hiç bir çalışmada aday arama alanını daraltmaya yönelik bir yaklaşıma rastlanmamıştır (Hameurlain et al. 2010; Çokuslu et al. 2012a).

Literatür taraması sonucu bulduğumuz eksiklikler bu araştırmanın çıkış noktasını oluşturmuştur. Büyük ağlarda arama alanını daraltmak amacıyla alt çizge (İng. subgraph) kullanımı bilinen ve uygulanan bir yöntemdir. Bu nedenle önerilen bazı algoritmalarda kullanılmak üzere, dağıtık sorgu sistemlerine yönelik bir alt çizge belirleme algoritması önerilmiştir. Bu sayede dağıtık sorgulama sitemlerinde, *i.* aday belirleme aşamasındaki işlemlerin azaltılması, *ii.* hızlandırılması, *iii.* önerilen algoritmalara ölçeklenebilirlik (İng. scalability) kazandırılması ve *iv.* daha önemlisi doğru adayların seçilmesi amaçlanmıştır. Daha önceki çalışmaların çoğuna ait maliyet modellerinde, ağdaki düğümlerin ilingesel özelliklerine yönelik bir parametreye rastlanmadığı için ve bu özelliklerin etkilerini araştırmak amacıyla, merkezilik (İng. centrality) tabanlı üç tane yeni kaynak atama algoritması önerilmiştir. Algoritmalar, adayların seçiminde verinin transfer edileceği düğümlerin birbirlerine göre konumunu, yakınlıklarını ve verinin transferi sırasında geçeceği patikaları dikkate almaktadır.

Bu yaklaşımla, belirlenen adayların sorgu işleme sisteminde kullanılması sonucu: *i.* Verinin iletim süresinin kısaltılması, *ii.* Veri trafiğinin azaltılması, *iii.* Veri trafiğinin ağın belirli bir bölgesi dışına çıkmaması ve *iv.* en önemlisi, doğru adayların seçilmesi amaçlanmıştır.

1.3 Benzetim Ortamında Doğrulama

Bu çalışmanın amacı, sistemindeki düğümlerin ilingesel konumlarının sorgu işleme performansı üzerindeki etkisini araştırmakır. Olabildiğince gerçek hayat şartlarından uzaklaşmadan, bu etkiyi ön plana çıkarabilmek ve var olan imkânlar doğrultusunda sağlıklı testler yapabilmek için aşağıdaki kabullenmeler yapılmıştır.

Varsayımlar:

i. Sistemin içinde bulunduğu çizgenin içerdiği düğüm sayısı bilinmemektedir. İşlemleri basitleştirmek amacıyla veri kaynağı listesindeki ilk düğüm, sorgu sahibi düğüm ve aynı zamanda başlatıcı düğüm olarak kabul edilmiştir.

ii. Çizgedeki tüm düğümler, her türlü görevi yerine getirebilecek düzeyde kaynağa sahip düğümlerdir.

iii. Çizge bağlı (İng. connected) ve yönsüzdür.

iv. Çizgedeki her kenar, sıfırdan büyük ve farklı olabilen ağırlık değerlerine sahiptir, yani çizge ağırlıklıdır (İng. weighted graph). Bir kenara ait ağırlık değerinin tersi, o kenarın kanal genişliği olarak kullanılmıştır. Bu yaklaşımla, aynı uzaklıkların aynı sürede kat edilmesi, ağırlıklı yönlendirme protokolü (İng. weighted routing protocol) ile uyum ve taşırma (İng. flooding) yöntemi ile en kısa yolların belirlenmesi sağlanmıştır (Bkz. Lemma-1.2).

v. Veri kaynağı olarak kullanılacak düğümlerin sayısı, kimlikleri (İng. id) ve birbirleri arasındaki en büyük uzaklık bilinmektedir. Bu bilgi sorgu işleminin, kaynak keşfi (İng. resource discovery) aşamasında elde edilebilmektedir (Hameurlain et al. 2010).

vi. Her düğüm kendi düğüm numarasını (İng. node id), var olan kenarlarının ağırlık değerlerini ve komşularının düğüm numaralarını bilmektedir.

vii. Merkezi İşlem Birimi (İng. Central Processing Unit, *CPU*) ihtiyacı ve yük dengeleme gibi kıstaslar dikkate alınmayacaktır.

viii. Belirlenen alt çizgenin ilingesi (İng. topology), en azından ilgili sorgu bitene kadar değişmeyecektir.

ix. İletişim sırasındaki bilgi işleme maliyeti, iletişim maliyetinden çok daha küçük olacağı için (İng. computation overheads) ihmal edilmiştir.

x. Belirlenen adayların performanslarını ölçmek amacıyla yapılan sorgu işleme benzetimi (İng. simulation) anında *TCP* ve ağırlıklı yönlendirme IP (İng. weighted routing) protokolü kullanılmıştır. Bu yaklaşımla, gerçek hayat şartlarının elde edilmesi amaçlanmıştır.

xi. Veri kaynaklarındaki veri tabloları yatay olarak bölünmüş ve başka bir düğümde kopyası bulunmamaktadır.

xii. Testi yapılan sorgunun sadece bir tane veri birleştirme operatörüne ihtiyacı olacağı kabul edilmiştir.

xiii. Algoritmaların belirlediği adayları kıyaslarken, eşitliği sağlamak ve testleri basitleştirmek adına her test için sabit 20 adet veri kaynağı (Çokuslu 2012c) olduğu kabul edilmiş ve bunlar ana çizgeden rastgele seçilmiştir.

Tanımlanan varsayımlardan da anlaşılacağı gibi, sorgu işleme sisteminin sadece kaynak atama aşamasına odaklanılmıştır. Aday belirleme işlemlerinde ise ilingesel özelliklerin etkisi vurgulanmak istenildiği için ilingesel özellikler hariç diğer parametrelerin, tüm düğümler için eşit olduğu kabullenilmiştir. Uygulama testi için sorgu işleme aşamasındaki sadece veri birleştirme aşaması benzetimlenmiştir. Bu benzetimde belirlenen adayların performans farklarını ön plana çıkartabilmek amacıyla sadece bir birleştirme operatörünün olduğu kabul edilmiştir. Kullanılan benzetim ortamının, tez çalışmasının amacı doğrultusunda gerçek hayat şartlarına uygun olması için özen gösterilmiştir.

Test ve benzetim ortamı olarak yaygın kullanımı olması, akademik çevrelerce kabul görmesi ve açık kaynak kodlu olması nedeniyle *NS3* (Network

Simulator 3) ağ benzeştiricisi kullanılmıştır (Nsnam 2008; K Sheshadri 2016; Sottile 2016). *NS3* pek çok protokol ve olayı destekleyen genel amaçlı bir ayrık olay benzeştiricisidir (İng. discrete event simulator). Çalışmamızda, *NS3*'ün bazı hazır uygulamaları kullanılmakla birlikte ihtiyaç duyulan bazı uygulamalar da geliştirilerek kullanılmıştır.

Bu tez çalışması özellikle büyük ölçekli sorgulama sistemlerine yöneliktir. Bu denli büyük ölçekli sorgulama sistemleri bildiğimiz kadarı ile örgü hesaplama (İng. grid computing) sistemlerinde mevcuttur. Örgü hesaplama sistemleri ise internet ağında bulunan bir alt sistemdir (Travieso et al. 2013). Dolayısı ile çizge modeli olarak internet ağ modeli örnek alınmıştır. İnternet ağı genel olarak küçük dünya (İng. small world), güç yasası (İng. power law), ölçeksiz (İng. scale free) ve karmaşık ağ (İng. complex network) özellikleri göstermektedir (Wang 2003). Bu özelliklerde ve istenilen boyutlarda örnek çizge verisi üretebilmek için *BRITE* adı verilen ilinge üreteci seçilmiştir (Medina et al. 2001). Bu ürün akademik çevrelerce kabul görmüş, uzun yıllardır kullanılan, açık kaynak kodlu ve en önemlisi, istenilen özelliklerde çizge üretebilen bir yazılımdır.

Yukarıdaki kabuller doğrultusunda bir tane alt çizge belirleme algoritması ve üç tane aday belirleme algoritması geliştirilmiştir. İki tane kıyas algoritması kodlanmıştır ve bir tane sorgu birleştirme benzetim algoritması tasarlanmıştır. Kıyaslama algoritmaları, önerilen algoritmaların performansını kıyaslamak amacıyla ve daha önceki çalışmalara dayandırılarak kodlanmıştır. Sorgu birleştirme benzetim algoritması ise belirlenen adayların başarılarını ölçmek amacıyla kodlanmıştır. Bu algoritma; sonuçların tarafsızlığını ve gerçek hayata uygunluğunu sağlamak amacıyla *NS3* geliştiricileri tarafından hazırlanmış standart bir uygulama olan "*TCP* büyük veri transfer" uygulaması kullanılarak gerçekleştirilmiştir.

1.4 Katkılar

Bu tez çalışmasında, dağıtık sorgu işleme sisteminin ara aşamalarından olan ve kaynak atama olarak adlandırılan adım için karmaşık ağlar üzerinde çalışabilecek algoritma önerilerinde bulunulmuştur. Yapılan çalışma bir eniyileme işlemidir ve polinom zamanda çözümü olmadığı ispatlanmış bir probleme aittir. Bu nedenle sezgisel bir yaklaşımla, en iyiye yakın (İng. near optimal) ve merkezilik tabanlı üç adet algoritma önerilmiştir. Bu algoritmaların hepsi, düğüm sayısı bilinmeyen bir ağda ve sınırları veri kaynağı düğümlerce belirlenen bir alt

çizge içerisinde çalışmaktadır. İki tanesi ise aday belirleme işlemine başlamadan önce, aday belirleme alanını daha da daraltmak amacıyla bir alt çizge belirleme algoritması çalıştırmaktadır. Bu sayede; *i.* seçilen veri kaynaklarının sayısı ve ağ üzerindeki yerleşimlerine bağlı olarak işlem maliyeti düşürülmüştür, *ii*. veri trafiğinin ağın sadece ilgili bölümünde kalması sağlanmıştır, *iii.* sistemin ölçeklenebilirliği arttırılmıştır ve *iv*. en önemlisi, tüm çizge yerine sadece ilgili düğümler arasından (alt çizge içinden) uygun adayları seçerek, daha doğru adayların daha hızlı belirlenmesi sağlanmıştır. Ana çizge ne kadar büyük olursa olsun, algoritmalar sadece alt çizge tarafından tanımlanan alan içerisinde çalışmaktadır. Bu sayede tüm çizgedeki düğümlerin en iyisi yerine alt çizge içerisindeki düğümlerin en iyileri aday olarak seçilmektedir. Çünkü tüm çizgenin en iyi adayı sorgu işleme için en uygun aday olmayabilir. Özellikle ilingesel açıdan değerlendirildiğinde; birbirine yakın düğümler arasındaki iletişim maliyeti daha düşük olurken, alt çizge içinde bulunmayan ama en büyük kanal genişliğine sahip düğümün iletişim maliyeti çok daha büyük olabilmektedir.

Sonuç olarak, önerilen algoritmalar ile üç farklı merkezilik yaklaşımı değerlendirilmiş ve aşağıdaki katkılar elde edilmiştir; *i.* Özellikle yakınlık merkezilik yaklaşımının sorgu sürecine ait işlem süresinin kısalması ve iletişim maliyetinin düşmesinde oldukça etkili olduğu görülmüştür, *ii.* Lemma-1.1 de detaylı olarak açıklanan ve bildiğimiz kadarı ile ilk defa uygulanan sınırlı merkezilik hesaplama yaklaşımı ile doğru adayların seçimini sağlamış ve algoritmaların maliyetlerini düşürmüştür, *iii.* Bildiğimiz kadarı ile dağıtık merkezilik algoritmaları üzerine yapılan ilk tarama ve sınıflandırma çalışması yapılmıştır, *iv.* İlingesel özelliklerin, sorgu işleme sisteminin veri birleştirme adayları seçiminde etkili bir ölçüt olduğu ispat edilmiştir. Bu nedenle sorgu işleme maliyet modeli tanımlanırken, bu modelin içine düğümün ilingesel özelliklerini içeren (merkezilik, yakınlık vb.) yeni bir parametrenin eklenmesinin uygun olacağı sonucuna varılmıştır.

1.5 Tanımlamalar

Bu çalışma birtakım protokoller üzerinden birbirleri ile iletişim kurabilen aygıtların oluşturduğu ağlar üzerinde yapılmıştır. Ağdaki aygıtlar düğüm olarak adlandırılmaktadır. Bildiğimiz kadarı ile bu tür ilişkisel yapıların analizinde en uygun yöntem çizge teorisidir ve 1948'lerde bu yöntemi ilk uygulayanlardan birisi Bavels'dir (Jain et al. 1999; Tasoulis and Vrahatis 2005). Dağitik sistemlerde

çeşitli problemleri çözmek icin çizge teorik algoritmalarin kullanılması yaygındır(Kavalci et al. 2014; Can Umut et al. 2016).

Bir çizge $G=(V,E)$ ise burada V boş olmayan düğümler kümesini ve E de boş olmayan kenarlar kümesini temsil etmektedir. Her $u, v \in V$ bir düğümü tanımlarken, her $e = (u, v) \in E$ de bir kenarı tanımlamaktadır, n çizgedeki düğüm sayısını temsil etmektedir ve $n=|V|$ dir, m ise çizgedeki kenar sayısını temsil etmektedir ve $m=|E|$ dir. Bir düğüme ait komşuluk listesi $\Gamma(v)$ ile çizgedeki en büyük derece Δ ile gösterilmiştir. Yürüyüş (İng. walk), $w(v_1, e_1, v_2, e_2, \ldots v_n, e_n,)$; v_1 ile v_n düğümleri arasında, sırasıyla e_1 den e_n ' e kadar olan kenarların kullanılması ile yapılan iletişimdir (Erciyes 2013).

Patika(İng. trail)**:** Yürüyüş anında bir kenarın birden fazla kullanılmaması durumudur. Yol (İng. path), $p(u, v)$ ise yürüyüş anında aynı kenar ve düğümün birden fazla kullanılmaması durumudur. Yol uzunluğu, $|p(u, v)|$ ile gösterilmektedir ve u ve v düğümleri arasındaki kenar sayısına eşittir (Erciyes 2013).

Uzaklık (İng. distance)**:** İki düğüm arasındaki en kısa yolun uzunluğudur, *distance(u, v)* fonksiyonu u ile v düğümleri arasındaki en kısa yolu tanımlar (Erciyes 2014). Uzaklık, iki düğüm arasındaki toplam kenar sayısını temsil edebileceği gibi kenarlar üzerinde tanımlanan ağırlıkları da temsil edebilir (Erciyes 2013).

Çap(İng. diameter)**:** D_G bir çizgedeki tüm düğümler içinde, herhangi iki düğüm arasında bulunan en büyük uzaklıktır (Erciyes 2014).

Dışmerkezlik: Bir v gibi bir düğümün dışmerkezlik (İng. eccentricity) değeri, *eccentricity(v)* fonksiyonu ile temsil edilir ve o düğümün çizgedeki diğer tüm düğümlere olan uzaklık değerleri içinden en büyük olanıdır (Erciyes 2014).

Yarıçap: Bir G çizgesinin yarıçapı (İng. radius), *radius(G)* fonksiyonu ile temsil edilir ve o çizgedeki tüm düğümlerden en küçük dışmerkezlik değerine sahip olan düğümün dışmerkezlik değerine eşittir (Erciyes 2013).

Düğüm Tipleri:

Vo: Sıradan çizge düğümleridir, $Vo \in V$ ve başlangıç anında tüm düğümler bu tiptedir.

Vc: Veri kaynaklarını temsil eden düğümlerdir (İng. data resource/colored node), $Vc \in V$ ve başlangıçta sadece başlatıcı düğüm (İng. initiator node) *Vi* tarafından bilinmektedirler. *VC* ise bu düğümlere ait kümenin adıdır.

Vuc: Alt çizge içinde bulunan ama veri kaynağı olmayan sıradan düğümlerdir (İng. uncolored node), $Vuc \in V$. Tüm *Vc* düğümlere olan uzaklıları *Vc* çapından küçük ya da eşit olan ve *Vc* düğüm olmayan düğümlerden oluşur. Bu tür düğümlere ait kümenin adı ise *VUC* dir.

$$VUC(v) = \{v \in V : \forall\, distance(v, Vc) \leq Vc\ diameter\} \backslash VC \qquad (1.1)$$

Vi: Başlatıcı düğüm (İng. initiator node), $Vi \in Vc$. Bu çalışmada sorgu sahibi (İng. query owner) aynı zamanda başlatıcı düğüm olarak kabul edilmiştir.

Alt Çizge: Bir *G* çizgesindeki tüm *Vc* düğümlere olan uzaklıları *Vc* çapından küçük ya da eşit olan düğümler kümesidir. Bu tür düğümlere ait kümenin adı ise *SG* dir.

$$SG = VC + VUC \qquad (1.2)$$

İndirgenmiş Alt Çizge: Bir *SG* alt çizgesinde bulunan düğümlerin, *Vc* düğüm yakınlık ortalamasına eşit ya da daha küçük yakınlık merkezilik değerine sahip olan *Vuc* düğümleri ile *Vc* düğümlerin tamamından oluşmaktadır. Bu tür düğümlere ait küme *RSG* olarak adlandırılmıştır. Alt çizgedeki her bir düğüme ait V_{cc} değeri, alt çizge belirleme algoritması tarafından hesaplanan yakınlık merkezilik değeridir. Vc_{cca} ise *Vc* düğümlere ait yakınlık merkezilik değerlerinin ortalamasıdır.

$$Vc_{cca} = \frac{\sum_{v \in VC} Vc_{cc}(v)}{|VC|}$$

$$RSG(v) = \{v \in VUC : \forall V_{cc} \leq Vc_{cca}\} + VC \qquad (1.3)$$

$A[n,n]$, çizgenin komşuluk (İng. adjacency) matrisidir ve her bağlantı/kenar 1 ile temsil edilmektedir (Erciyes 2013).

σ_{st} , *s* ve *t* düğümleri arasındaki kısa yol sayısını göstermektedir. $\sigma_{st}(v)$ fonksiyonu ise *s, t* ve *v* birbirlerine eşit değilken, *s ile t* arasındaki kısa yollarının kaç tanesinin *v* düğümü üzerinden geçtiğini göstermektedir. $\sigma_{st}(u,v)$ fonksiyonu ise *s, t* ve *v* birbirlerine eşit değilken, *s ile t* arasındaki kısa yollarının kaç tanesinin e(*u,v*) kenarı üzerinden geçtiğini göstermektedir (Merrer and Trédan 2009).

$P_{st}(v)$, *s* ve *t* düğümleri arasındaki kısa yolların bir *v* düğümü üzerinde geçme olasılığını göstermektedir.

$$P_{st}(v) = \frac{\sigma_{st}(v)}{\sigma_{st}} \qquad (1.4)$$

$P_{st}(u,v)$, *s* ve *t* düğümleri arasındaki kısa yolların, bir e(*u,v*) kenarı üzerinde geçme olasılığını göstermektedir.

$$P_{st}(u,v) = \frac{\sigma_{st}(u,v)}{\sigma_{st}} \qquad (1.5)$$

1.6 Tezin Yapısı

Tezin bundan sonraki bölümleri şu şekildedir. Bölüm 2 de merkezilik türleri ve var olan dağıtık merkezilik tabanlı algoritmalar üzerine yapılan bir tarama çalışması sunulmuştur. Tarama sonucu elde edilen veriler ışığında; *i.* Algoritmaların temel özellikleri dikkate alınarak bir sınıflandırma çalışması yapılmıştır, *ii.* Bir takım nicel ve nitel kısıtlar dikkate alınarak algoritmalar analiz edilmiş ve karmaşık ağlar için uygunluğuna yönelik değerlendirmelere yer verilmiştir. Bölüm 3 de tarama çalışması sonucu elde edinilen bilgi ve sonuçlar doğrultusunda belirlenen; *i.* Problemin çözümüne yönelik önerdiğimiz 3 algoritmaya, *ii.* Önerilen algoritmaların kıyaslanması amacıyla kodlanan, geçmişteki çalışmalara ait iki kıyas algoritmasına ve *iii.* Belirlenen adayların performanslarını test edebilmek amacıyla hazırlanan ve sorgu işleme sürecindeki veri birleştirme aşmasını benzetimleyen bir benzetim algoritmasına yer verilmiştir. Kullanılan algoritmalar hakkında detaylı bilgi verilip teorik analizleri yapılmıştır. Bölüm 4 de ise önerilen ve kıyas algoritmalarının benzetim ortamında çalıştırılmasıyla elde edilen sonuçlara yer verilmiştir. Benzetim ortamında elde edilen pratik sonuçlar hakkında değerlendirmeler yapılmıştır. Son olarak bölüm 5

de ise teorik ve pratik sonuçlar yorumlanıp, tez çalışması değerlendirilmiş ve gelecekteki çalışmalar hakkında bilgi verilmiştir.

2. İLGİLİ ÇALIŞMALAR

2.1 Merkezilik Tabanlı Dağıtık Algoritmalar

2.1.1 Giriş

Merkezilik ölçütü, farklı adlarda da olsa araştırmacılar tarafından uzun zamandır ilgilen bir konudur. Günümüzde ağlarının giderek daha da karmaşıklaşması ve büyük miktarlardaki veri analizlerine olan ihtiyaç gibi nedenlerden dolayı merkezilik ölçütlerine olan ilgi bu günlerde daha da artmıştır. Merkezilik ölçütleri, genellikle bir ağda göreceli olarak önemli olan üyeyi belirlemek için kullanılır. Örneğin, bir bilgisayar ağında en yüksek dereceli düğüme ait merkezilik değeri, yüksek iletişim potansiyelinden dolayı diğerlerinden daha yüksek olabilir. Merkezilik ölçütlerinin bilgisayar ağlarında olduğu gibi sosyal ağlarda, kablosuz ağlarda, suçlu ağlarında (İng. criminal networks) liderin belirlenmesinde, büyük veri analizlerinde, örüntü hesaplama (İng. grid computing) ve askeri uygulamalar gibi birçok alanda kullanımı mümkündür (Wehmuth and Ziviani 2012b).

Günümüzde, tanımlanmış birçok merkezilik ölçütü bulunmaktadır. Bunların çoğu ortalama uzaklık, kısa yollar, nesneler arasındaki ilişkiler ve nesneler arasındaki trafik ya da veri akışı gibi kısıtları kullanarak değerlendirme yapmaktadır (Wehmuth et al. 2011a). Yapılacak çalışmanın özelliklerine göre bir uygulamada, merkezilik ölçütlerinin bir ya da birden fazlası aynı anda kullanılabilmektedir (Marsden 2002). Bununla birlikte, bildiğimiz kadarı ile hangi merkezilik ölçütünün hangi uygulama alanında kullanılması gerektiğine yönelik tanımlanmış belirli bir kural yoktur. Farklı merkezilik ölçütlerinin belirli uygulama alanları üzerindeki etkileri, araştırılmakta olan konulardan biridir. Bilinen merkezilik ölçütlerinin sınıflandırılması, etkilerinin daha iyi anlaşılabilmesi açısından bu alanda yapılacak çalışmaların ilk adımlardan biri olabilir. Bildiğimiz kadarı ile ilk sınıflandırma Linton C. Freeman tarafından yapılmıştır (Freeman 1979). O yıllara ait, yayınlanan merkezilik ölçütlerini değerlendirmiştir. Merkezilik ölçütlerini derece (İng. degree), yakınlık (İng. closeness) ve aradalılık (İng. betweenness) merkezilik ölçütleri olarak üç temel sınıfa ayırmıştır. Çalışmasının bir sonucu olarak, bu üç tür merkezilik ölçütünün sadece yıldız ilingesine sahip ağlarda (İng. star networks) aynı nesneyi önemli olarak bulduklarını belirtmiştir (Freeman 1979). Diğer bir deyişle diğer ağ

ilingelerinde, genellikle her merkezilik ölçütü farklı bir nesneyi önemli olarak bulmuştur.

Merrer ve arkadaşı ise bilinen merkezilik türleri üzerine yaptıkları çalışmada ilginç çıkarımlarda bulunmuşlardır (Merrer and Trédan 2009). Farklı merkezilik türlerinin çizgelerin farklı özellikleri ile ilgilenmesinden dolayı aynı çizge üzerinde faklı sonuçlar üretebileceğini doğrulamışlardır. Dahası, bazı merkezilik türlerinin ürettiği sonuçlar arasında bir ilişki (İng. correlation) olmadığını göstermişlerdir. Eşmerkezlilik (İng. eccentricity), derece (İng. degree), özvektor (İng eigenvector) ve yakınlık (closeness) merkeziliklerinin çizge bağlantılılığı (İng. connectivity) ile ilgili önemli bilgiler vermediğini gözlemlemişlerdir. Bir başka çıkarımları ise merkezi düğümün çizgenin bağlantılılığı üzerine olan etkisi hakkındadır. Bunu test etmek için farklı merkezilik türlerini kullanarak, sırasıyla en merkezi düğümleri çizgeden çıkarmışlardır. Sonuç olarak, klasik aradalılık merkezilik ile rastgele yürüyüş (İng. random walk) merkezilik türlerinin çizgedeki kesim noktaları ya da köprü konumlarını diğer merkezilik türlerinden daha hızlı bulduğunu göstermişleridir. Ayrıca rastgele yürüyüş merkezilik türünün, klasik aradalılık merkezilik türüne göre çok daha iyi sonuçlar verdiğini göstermişlerdir. Çünkü rastgele yürüyüş merkezilik türü, sadece en kısa yolları değil olası tüm yolları değerlendirmektedir. Son çıkarımları ise merkezi düğümün çizgenin ortalama çapına (İng. average route) olan etkisi üzerinedir. Bunu test etmek amacıyla, farklı merkezilik türlerini kullanarak, sırasıyla en merkezi düğümleri çizgeden çıkarmışlar ve ortalama çapı tekrar hesaplamışlardır. Sonuç olarak, özellikle derece merkezilik türünün çizge çapının hızla büyümesinde önemli bir etkisi olduğunu bulmuşlardır (Merrer and Trédan 2009).

Hızla gelişen ağ ve bilgisayar teknolojileri, günümüzde çok büyük ve karmaşık ağların oluşmasına neden olmuştur. Ölçüsüz (İng. scale free) olarak tanımlanan bu ağlar milyonlarca düğüm ve milyarlarca kenardan oluşabilmekte ve giderek büyümektedirler (Erciyes 2014). Bildiğimiz kadarı ile var olan merkezilik algoritmaları polinomsal zamanda çalışmaktadır. Fakat düğüm sayısının büyüklüğünden dolayı merkezi (İng. central) ve paralel algoritmalar donanım sınırlarını zorlar hale gelmiştir. Dolayısı ile merkezilik algoritmalarının karmaşıklıkları önem kazanmıştır. Bu probleme bir çözüm olarak dağıtık merkezilik algoritmaları düşünülmektedir. Çünkü ağda bulunan her bir birimin işlem yapabilen bir birim olarak kabul edildiği dağıtık sistemlerde, algoritmaların zaman ve bellek karmaşıklıkları önemli ölçüde azaltılabilmektedir.

Bildiğimiz kadarı ile dağıtık merkezilik algoritmaları üzerine yapılmış bir araştırma çalışması bulunmamaktadır. Dahası, dağıtık merkezilik algoritmaları üzerine çok az sayıda çalışma bulunabilmiştir (Kermarrec et al. 2011).

Bu bölümde, dağıtık merkezilik algoritmaları üzerine yapılan bir araştırma ve bu algoritmalar için yeni bir sınıflandırma çalışması sunulmuştur. Merkezilik ölçütlerinin temel amaç ve özellikleri dikkate alınarak, bir sınıflandırılma yapılmıştır. Bu sınıflar; bilinilirlik (İng. popularity), erişilebilirlik (İng. accessibility) ve geçirgenlik (İng. permeability) olarak adlandırılmıştır.

Bilinirlik sınıfındaki merkezilik türlerinde amaç, en çok bilinen/tanınan düğümü bulmaktır. Bu sınıfta, bir düğümün bilinilirlik değerinin, çizgedeki derecesi (komşu sayısıyla) ile ilgili olduğu kabul edilmiştir. Örneğin, derecesi yüksek olan düğümler diğerlerinden daha bilinir düğümlerdir. Derece; en iyi bilinen, basit ve çok kullanılan ilingesel bir parametredir. Derece ve özvektor merkezilik türleri bu sınıfın en iyi bilinen örneklerindendir (Borbash et al. 2007; Ilyas et al. 2011; Noori 2011; Montresor et al. 2013).

Erişilebilirlik sınıfındaki merkezilik türlerinde amaç, tüm düğümlere en kısa yoldan ulaşabilen diğer bir ifadeyle en yakın düğümleri bulmaktır. Temel parametre düğümler arası uzaklıklardır. Herhangi iki düğüm arasındaki uzaklık, bu iki düğüm arasındaki en kısa yol olarak kabul edilmiştir. Bu sınıftaki merkezilik türleri, tüm iletişimin sadece kısa yollar üzerinden yapıldığını kabul etmektedir. Yakınlık merkezilik türü bu sınıfın en iyi bilinen örneklerinden biridir. Bir düğümün yakınlık merkezilik değerini hesaplamak için ilgili düğümden diğer düğümlere olan uzaklıklarının toplamının bulunması gerekmektedir. Sonuç olarak, yakınlık değeri en küçük olan düğüm diğer düğümlere en yakın düğüm olarak seçilmektedir (Lehmann and Kaufmann 2003; Xing et al. 2010; Wehmuth and Ziviani 2011b; Wehmuth 2012a; Wehmuth and Ziviani 2012b; ChongGun and Mary 2013).

Geçirgenlik sınıfı ise bir düğümün üzerinden geçirdiği potansiyel trafik yükünü dikkate almaktadır. Geçirgenlik değeri yüksek düğümler, ağdaki köprü veya tıkanma noktalarını gösteren en uygun aday düğümler olarak düşünülebilir. Bildiğimiz kadarı ile bir düğüme ait geçirgenlik değeri iki temel yaklaşımla hesaplanabilmektedir. İlk yaklaşım kısa yolları kullanır ve ağdaki iletişimin sadece kısa yollar üzerinden yapıldığını kabul eder. Bu yaklaşımı kullanan algoritmalar kararlı (İng. deterministic) bir yapıya sahiptir. Diğer yaklaşım

rastgele yürüyüş (İng. random walk) ve benzeri yöntemleri kullanır ve iletişim için olası tüm yollar kullanılabilir. Aradalılık (İng. betweenness) merkezilik (kenar ve düğüm aradalılık merkezilik), gerginlik (İng. stress) ve rastgele yürüyüş aradalılık merkezilik türleri bu sınıfın en iyi bilinen örneklerinden bazılarıdır. Bu sınıfa ait merkezilik türleri, bir ağdaki küme sınırlarını bulmak (İng. cluster border), tıkanıklık (İng. congestion control) noktalarını bulmak ve kesim noktalarını (İng. cut point) bulmak gibi pek çok amaçla kullanılmaktadır (Lehmann and Kaufmann 2003; P. Fekete et al. 2005; Nanda and Kotz 2008; Oliveira et al. 2010; Bradler et al. 2011; Kermarrec et al. 2011; Guidi et al. 2014; Lulli et al. 2015; Hua et al. 2016).

Bu araştırmanın temel amaçlarından biri, sorgu işleme sistemlerinin aday belirleme aşamasında, dağıtık merkezilik yöntemlerini kullanarak düğümlerin ilingesel özelliklerinin etkisini araştırmaktır. Bu amaçla geliştirilen algoritmalar çok miktarda veri transferi gerektiren sorgu işleme sistemlerinde ve internet gibi büyük ve karmaşık ağlar üzerinde çalışacaktır. Bu nedenle daha önce bu alanda yapılan çalışmaları değerlendirmek amacıyla, karmaşık ağların analizine yönelik bir takım kısıtlar belirlenmiştir. Büyük ve karmaşık ağların davranışlarının modellenmesi için uzun zamandır çalışılmaktadır. Bu çalışmaların sonucunda, bu tür ağların belirlenebilen temel özellikleri: *i.* Boyut sınırı olmamak (İng. scale free), *ii.* Karmaşıklık, *iii.* Dinamiklik, *iv.* Güç yasasına (İng. power law) göre büyümek, *v.* Düşük ortalama çizge çapına sahip olmak, *vi.* Yüksek gruplanabilirlik (İng. clustering coefficient) özelliği göstermek ve *vii.* Yaşlanabilirliktir (Erciyes 2014). Bu özellikler dikkate alınarak, daha önce yapılan çalışmalar ise: *i.* Karmaşıklık (İng. complexity), *ii.* Ölçeklenebilirlik (İng. scalability), *iii.* Dinamiklik (İng. dynamicity) ve *iv.* Güvenilirlik (reliability) açısından değerlendirilmiştir.

Karmaşıklık, algoritmalarının performanslarının kıyaslanmasında kullanılan temel kısıtlardan biridir. Bu çalışmada sadece mesaj ve zaman karmaşıklıkları ele alınarak değerlendirme yapılmıştır. Ölçeklenebilirlik, özellikle hızla büyüme olasılığı olan sistemlerde çalışacak algoritmalar için anlamlı bir ölçüttür. İlgili çalışmanın, sistemin büyüme oranına göre performansının nasıl değişeceğini tanımlamak için kullanılmaktadır. Bu çalışmada karmaşıklığı $O(n^3)$ ya da daha küçük olan algoritmalar ölçeklenebilir kabul edilmiştir. Dinamiklik ise algoritmanın, yapısı/ilingesi değişebilen ağlar üzerinde kullanılabilirliğini tanımlar. Örneğin hareketli (İng. mobile) ve sosyal ağlar son derece dinamik ağlardır. Böyle bir ağda meydana gelen her değişiklikte, tüm düğümlere ait

merkezilik değerlerini tekrar tekrar hesaplamak yerine, bu değişiklikten sadece etkilenen düğümlere ait yeni değerleri hesaplayabilmek algoritmaya dinamiklik kazandıracaktır. Bu çalışmada algoritmaların dinamiklik özelliğinin olup olmadığına bakılarak bir değerlendirme yapılmıştır. En temel ölçütlerden biri olan güvenilirlik, algoritmanın ürettiği sonuçların doğruluk derecesini ya da kullanılabilirliğini belirlemek için kullanılmaktadır. Bazı algoritmalar daha iyi performans elde edebilmek amacıyla, tüm çizge verisini kullanmak yerine yerel ya da bir kısım çizge verisini kullanarak sonuç üretebilmektedir. Bu gibi durumlarda sonuçların doğruluk oranı ya da hassasiyeti değişebilmekte, merkezilik değerlerindeki sıralamalar farklı olabilmektedir. Bu çalışmada algoritmaların güvenilir sonuçlar üretip üretmediğine de bakılarak değerlendirme yapılmıştır.

Sonuç olarak, bulabildiğimiz kadarı ile merkezilik ölçütlerini kullanan dağıtık algoritmalar incelenmiştir. Çalışmamızın bu bölümünde, sağladığımız en önemli katkı, bu çalışmanın bildiğimiz kadarı ile dağıtık merkezilik algoritmaları üzerine yapılan ilk araştırma çalışması olmasıdır. Bu çalışmada dağıtık merkezilik tabanlı algoritmalar üzerine detaylı bir araştırma yapılmıştır. Bulunan algoritmalar bir takım kısıtlar üzerinden kıyaslanmış ve sınıflandırılmıştır. Son olarak algoritmalar hakkında genel bir değerlendirme yapılmış, zayıf ve güçlü özellikleri belirtilmiştir.

2.1.2 Dağıtık merkezilik algoritmaları

Bu bölümde dağıtık merkezilik algoritmaları incelenerek detaylı bir analiz ve sınıflandırma çalışması yapılmıştır. Her bir algoritma, daha önce belirlenen kısıtlar doğrultusunda değerlendirilmiştir. Her sınıfa ait bölümün sonunda ise genel bir değerlendirme yapılmıştır.

2.1.2.1 Bilinirlik sınıfı merkezilik algoritmaları

Bu sınıfta, bulunabilen çalışmalar üç temel merkezilik yaklaşımı dikkate alınarak incelenmiştir.

Derece Merkezilik (İng. Degree centrality) ($C_D(v)$): Oldukça basit bir merkezilik türüdür. Düğümlerin derecesi ya da komşu sayısı temel parametresidir. Ana fikir, derecesi yüksek olan düğümlerin diğerlerinden daha fazla iletişim yapabileceği üzerinedir. Bu yaklaşım arkadaşlık ölçütü olarak da yorumlanabilir. Bu nedenle yürüyüş tabanlı veri aktarım işleyişleri için oldukça uygun bir

yaklaşımdır. Örneğin hastalık veya dedikodu yayılımı, para değişim işlemi gibi (Borgatti 2005). Bildiğimiz kadarı ile ilk kez 1954'te Shaw tarafından geliştirilmiş ama modern şekli Freeman tarafından tanımlanmıştır (Freeman 1977; Freeman 1979).

Bir *G (V, E)* çizgesinde, bir *v* düğümünün, derece merkezilik değeri $C_D(v)$ ile temsil edilmektedir. Yönsüz çizgelerde $C_D(v)$, *v* nin derecesine eşittir. Ama yönlü çizgelerde bu değer giren kenarların sayısı (İng. in degree) ve çıkan kenarların sayısı (İng. out degree) olarak ayrı ayrı değerlendirilmelidir. Aşağıdaki denklem ile hesaplanabilmektedir.

$$C_D(v) = deg(v) = \sum_{j \in V} A[v, j] \qquad (2.1)$$

Komşuluk matrisi kullanan bir merkezi algoritma düşünüldüğünde, bir düğüm için zaman karmaşıklığı $O(n)$ dir. Çizgedeki düğüm derecelerini sıralamanın zaman maliyeti ise klasik yöntemler kullanıldığında $O(n^2)$ dir (Lehmann and Kaufmann 2003; Borgatti and Everett 2006; Zhuge and Zhang 2009; Wehmuth et al. 2011a; Nomikos George 2013; Erciyes 2014). Bu merkezilik algoritması ölçeklenebilirdir, ama dinamik özelliği yoktur ve güvenilirliği sınırlıdır. Çünkü bir çizgede aynı dereceye sahip çok sayıda düğüm olabilir ve derece parametresi sadece yerel ilişkileri yansıtmaktadır.

Anonim Dağıtık Derece Merkezilik (İng. Anonymous Distributed Degree Centrality) Algoritması (Borbash et al. 2007): İlk defa kim tarafından tanımlandığına dair bir bilgi bulunamamıştır. Derece merkezilik algoritması yapısı gereği, dağıtık olarak uygulanmaya çok uygundur (Freeman 1977). Bu yüzden pek çok çalışmada ya da ders notunda kolaylıkla bulunabilir. Komşu düğümler arası iletişimi temel aldığı için derece hesaplaması kolaylıkla değişik yollarla yapılabilmektedir. Örneğin, her düğümün taşırma yolu ile "merhaba" mesajı gönderdiği bir sistemde, bir düğümün aldığı mesaj sayısı o düğümün derecesine eşittir. Böyle bir yaklaşımın zaman karmaşıklığı $O(1)$ ve mesaj karmaşıklığı $O((n\Delta)$ dir. Günümüzde yapılan pek çok çalışmada her düğümün komşularını bildiği en baştan kabul edilmektedir.

Oldukça hızlı, kararlı, ölçeklenebilir ve kolay uygulanabilir bir algoritmadır. Ama dinamiklik yoktur ve merkezilik değerinin güvenilirliği sınırlıdır. Çünkü çok fazla sayıda aynı merkezilik değerine sahip aday üretebilmektedir.

Dağıtık Yığmalı Derece Merkezilik (İng. Distributed Cumulative Degree Centrality) ($C_{DCD}(V)$) Algoritması (Noori 2011): Noori tarafından sunulan ve pek çok türevi bulunan bir dağıtık derece merkezilik algoritmasıdır. Komşularının derecelerini kullanarak hesaplama yapan ve özvektor ile klasik derece merkezilik arası bir yaklaşımdır. Bir düğüme ait $C_{DCD}(v)$ değeri, o düğümün komşularının derecelerinin toplamına eşittir. Komşu sayısı ise algoritmaya bir parametre olarak verilen l parametresine bağlıdır. Hesaplama l kadar uzaklıktaki komşular dikkate alınarak yapılmaktadır. l, kullanıcı tarafından belirtilmekte ve bir ile çizge çapı arasında bir değer alabilmektedir. Noori, çalışmasında sadece bir takım denklemler vermiş ama algoritma karmaşıklıklarından bahsetmemiştir. l parametresinin değeri elde edilen merkezilik değerinin etkinliği açısından oldukça önemlidir.

$$\mathrm{C_{DCD}}(v) \sum_{\mathrm{j}=1}^{l} \sum_{i \epsilon \Gamma(v)_j} deg(i) \qquad (2.2)$$

Bildiğimiz kadarı ile bu yaklaşım zaman uyumsuz dağıtık bir algoritma ile gerçeklenebilir. Her düğüm kendi derece bilgisini, mesaj ömrünü (İng. Time To Live *TTL*) l ye eşitleyerek komşularına taşırma yöntemi ile mesaj gönderebilir. Bu mesajlar sayesinde, her düğüm kendi yığmalı merkezilik değerini hesaplayabilir. Bu durumda mesajlar l kadar atlayacağı için algoritmanın zaman karmaşıklığı $\Theta(l)$ ya da $O(D_G)$ kadar olacaktır. Her düğüm derecesini komşularına gönderecek ve bu mesaj l kadar atlayarak yayılacağı için algoritmanın mesaj karmaşıklığı $O(n\Delta l)$ dir. En kötü durumda bu algoritmanın mesaj karmaşıklığı $O(n\Delta D_G)$ olacaktır.

Algoritmanın karmaşıklığı ölçeklenebilir olarak kabul edilebilir. Çünkü Δ ve D_G değerleri, büyük ağlar düşünüldüğünde genellikle n den çok daha küçük değerler olacaktır. Algoritmada dinamiklik özelliği bulunmamaktadır. Güvenilirlik ise tamamen l parametresinin büyüklüğüne bağlıdır. l parametresi büyüdükçe, elde edilen sonuçlar yerellikten uzaklaşıp tüm çizgeyi kapsar hale gelmektedir. Bu yaklaşımın sonuçları; l değeri küçüldükçe derece merkezilik özelliği, büyüdükçe özvektor merkezilik özelliği göstermektedir (Noori 2011). Bu nedenle karalı bir algoritma olmakla birlikte, güvenilirliği sınırlı bir algoritmadır.

Özvektör Merkezilik (İng. Eigenvector Centrality) Algoritması: Bu merkezilik türünde, bir düğümün merkezilik değeri komşularının önemi ile hesaplanmaktadır. Bir başka değişle, eğer bir düğüm yüksek merkezilik değerine sahipse aynı zamanda önemli komşulara sahip demektir. Kısacası, bir düğümün

diğer tüm düğümlerin merkezilik değerini etkilediği, dolayısı ile tüm çizgeyi kapsayan genel bir yaklaşımdır ve yoğun matematiksel işlem gerektiren (İng. spectral) bir ölçüttür (Noori 2011). Merkezi algoritmalar komşuluk matrisini kullanarak özvektor merkezilik değerlerini hesaplayabilmektedir. Matematiksel yollarla olan çözümü ilk defa 1953 yılında Katz (Borgatti 2005) tarafından sunulmuş olup modern formu ise Bonacich tarafından tanıtılmıştır (Bonacich 1972).

Çizge *G(V,E)* iken *i* ve *j* düğümleri temsil etmektedir. x_i değişkeni ise i. düğümün merkezilik değerini temsil etmektedir. x_i aşağıdaki denklem ile hesaplanmaktadır (Phillip Bonacich 2001).

$$x_i = \frac{1}{\lambda_{max}} \sum_{j \in \Gamma(i)} x_i = \frac{1}{\lambda_{max}} \sum_{j \in V} A[i,j] x_j \qquad (2.3)$$

Γ(i), i' nin komşu kümesi ve λ ise özdeğere (İng. eigenvalue) ait, sıfıra eşit olmayan bir çapraz (İng. diagonal) matristir. λ_{max} ise λ'ların en büyüğüne ait değerdir. Vektörsel gösterimi aşağıdaki gibidir.

$$\vec{x} = \frac{1}{\lambda} A\vec{x}$$

$$\vec{x}\,\lambda = A\vec{x}$$

$$A\vec{x} - \vec{x}\,\lambda = 0$$

$$(A - \lambda I)\vec{x} = 0 \qquad (2.4)$$

Burada I birim matristir ve $\vec{x}$ ise A nin özvektör matrisidir. Ancak $det(A - \lambda I) = 0$ şartı sağlanmalıdır. Bu durumda A matrisinin karakteristik eşitliği elde edilmektedir (Erciyes 2014). Bu vektörün elemanları ise ilgili düğümün özvektör merkezilik değerlerini göstermektedir. En büyük değere sahip olan düğüm en merkezi düğüm demektir. Formül yardımıyla 1 ile *n* arasındaki sabit λ değerleri hesaplanmaktadır. $\vec{x}$ vektörü ise λ_{max} yardımıyla hesaplanmaktadır. Geleneksel, merkezi özvektör merkezilik algoritmasının zaman karmaşıklığı $O(n^3)$ dür. Fakat farklı matris çarpma yöntemleri kullanılarak bir takım iyileştirmeler yapılabilmektedir (Wehmuth et al. 2011a). Özvektör merkezilik türü, dağıtık olamayan haliyle ölçeklenebilirliğin sınırındadır ve dinamiklik özelliği yoktur ama güvenilirliği yüksektir. Çünkü bir düğümün merkezilik değeri üzerinde çizgedeki düğümlerin hepsinin etkisi bulunmaktadır.

Merkezi Olmayan Önemli Bileşen Merkezilik (İng. Decentralized Principal Component Centrality PCC) Algoritması (Ilyas et al. 2011): Ilyas ve arkadaşı, daha önceki çalışmalarının geliştirilmiş bir hali olarak *PCC* algoritmasını sunmuşlardır (Ilyas and Radha 2010). Bu çalışma sosyal ağlardaki en fazla *k* tane bilgi toplayıcı düğümün belirlenmesi (İng. identifying the top-*k* information hubs in a social network) problemi üzerinedir. *PC*C algoritmasında temel fikir: Bir düğümün kendi bağlantıları zayıf olsa bile eğer iyi bağlantılı komşulara sahipse ağ üzerinde merkezi bir konuma sahiptir, şeklindedir. Fikir, veri akışı üzerine kurulmuştur. Özvektör merkezilikte olduğu gibi düğümler arası bir etkileşim vardır. Ayrıca bu çalışmanın birden fazla grup içeren sosyal ağlar için tasarlanmış olmasından dolayı çok kutuplu bir yapıya sahip olduğu belirtilmiştir. Merkezi olmayan *PCC* algoritması, *Kempe-McSherry* algoritması (*KM*) (Kempe and McSherry 2008) kullanılarak gerçeklenmiştir. *KM* ise spectral analiz çalışmalarını dağıtık yapmak için kullanılan bir algoritmadır.

Ilyas ve arkadaşları, çalışmalarında yalancı kod (İng. pseudo code) vermemişler, test ya da benzetim ortamına yönelik bir açıklamada bulunmamışlardır. Onların verdiği açıklamalara göre, *KM* tabanlı algoritmalarının zaman karmaşıklığı $O(\tau_{mix}\ logn)$ dir. Burada *n* çizgedeki düğüm sayısını ve τ_{mix} ise "Markov chain" ile geçiş (İng. transition) matrisini birleştirme (İng. mixing) zamanıdır. Geçiş matrisinin ise komşuluk matrisinin satır-normalleştirilmiş (İng. row-normalized) hali olduğu belirtilmiştir. Yapılan açıklamalara göre, ölçeklenebilir bir çalışma olup yönlü ve yönsüz çizgeler üzerinde çalışabilmektedir. Fakat dinamiklik özelliğine sahip değildir. Sadece komşuluk ilişkilerini kullanarak çalıştığı ve oldukça düşük mesaj karmaşıklığına sahip olduğu belirtilmiştir. Yazarlara göre, düğümler her tekrarlamada (İng. iteration) komşularına bir mesaj göndermektedir. Bu durumda algoritmanın tekrarlama başına mesaj karmaşıklığı $O(in\Delta)$ dir. *i,* burada tekrar sayısını belirtmektedir. Ayrıca *PCC* algoritmasına ait sonuçların doğruluğunun tekrar sayısı ile doğru orantılı olduğu açıklanmıştır. Yazarların iddiasına göre *PCC* algoritması, gerçek dünyaya ait sosyal ağlar üzerinde 10-20 arasında özvektör ile kabul edilebilir sonuçlar üretebilmektedir. Ilyas ve arkadaşı çalışmalarını gerçek dünya verileri üzerinde test etmişler (6 milyon düğüm ve 40 milyon kenarı olan facebook ve twitter verileri ile). Sonuç olarak çalışmalarının önceki çalışmalardan %50 daha hassas sonuç elde eden ve kullanıcılara ait bilgileri koruyan bir yaklaşım olduğunu belirtmişlerdir.

***k*-Çekirdek Merkezilik (İng. *k*-Core Centrality) Algoritması:** *k-çekirdek* merkezilik bir tür ayrıştırma (İng. decomposition) tekniği olarak değerlendirilmektedir ve 1983 yılında Seidman tarafından önerilmiştir (Batagelj and Zaveršnik 2011). Ana fikir, birbirleri ile bağlantıları çok olan düğümlerden alt çizgeler oluşturmaktır. Genellikle büyük çizgeleri bölmek ve daha küçük ama yoğun alt çizgeler elde etmek için kullanılmaktadır.

k-çekirdeklilik (İng. *k-coreness*) mümkün olan en büyük oranda küçültülmüş alt çizgeleri belirlemekte kullanılan bir yöntemdir ve *k-cores ya da* k-*shells* olarak adlandırılabilmektedir. Basitçe, çizgeden sırayla, derecesi *k'*dan küçük olan düğümlerin çıkarılmasıdır. Bu işlem, *k*'dan küçük dereceli düğüm kalmayıncaya kadar sürdürüldüğünde, elinizde en düşük *k* dereceye sahip düğümlerden oluşan bağlantılı bir alt çizge oluşmaktadır.

k-coreness işlemi, *k-çekirdek ayrıştırma* (İng. *k-core decomposition)* olarak da adlandırılmaktadır. Bir düğümün çekirdeklilik değeri *k* ise bu düğüm en düşük derecesi *k* olan düğümlerden oluşan bir alt çizgenin içindedir. Bu şekilde bir çizgede farklı *k* değerlerine göre farklı alt çizgeler oluşturulabilmektedir. Çizge, en merkezinde *k* dereceli düğümlerden oluşan, onun etrafında ise *k-1* dereceli düğümlerden oluşan gibi, *k* tane farklı büyüklükteki alt çizgeye bölünebilmektedir. Örneğin bir *k-çekirdeklilik* alt çizgesindeki bir düğüm, *k-1-çekirdeklilik* alt çizgesinin de bir elamanı olabilir ama bunun tersi mümkün değildir. Bu yaklaşımda Batagelj–Zaverˇsnik algoritmasından esinlenilmiştir (Batagelj and Zaveršnik 2011).

Çizge *G (V, E)* iken, *k* önceden tanımlanmış bir tamsayıdır ve *k-çekirdek* alt çizgenin (H_k) tanımı aşağıdaki gibidir;

Alt çizge $H_k = (C, E|C), C \subseteq V$ dir. *C, k-çekirdekli* düğümlerden oluşan, indirgenmiş bir alt çizgedir. Ancak ve ancak $\forall v \in C : degH_k(v) \geq k$ iken H_k mümkün olan en fazla indirgenmiş alt çizgedir (Batagelj and Zaveršnik 2011).

Merkezi *k-çekirdek* algoritmasının zaman karmaşıklığı *O(m)* olarak verilmiştir (Montresor et al. 2013). Bu algoritma, ölçeklenebilirlik sınırları içinde olmakla birlikte dinamiklik özelliğine sahip değildir. Güvenilirlik açısından uygundur çünkü tüm çizge verisini kullanarak değerlendirme yapmaktadır. Ayrıca kararlı bir algoritmadır.

Dağıtık *k*-Çekirdek Ayrıştırma (İng. Distributed *k*-Core Decomposition) Algoritması (Montresor et al. 2013): Montresor ve arkadaşları tarafından yeni bir algoritma olarak sunulmuştur. Zaman uyumlu ve dağıtık bir algoritmadır. Hem bire bir (İng. one to one) hem de birden çokluya (İng. one to many) bağlantılı düğümler üzerinde uygulanabileceği belirtilmiştir. Algoritmaları, kendileri tarafından ispat edilen yerellik teoremine (İng. locality theorem) dayanmaktadır. Anladığımız kadarı ile yönlü ve yönsüz çizgelerde çalışabilen bu algoritma, yerel ağ verilerini kullanarak *k-çekirdekli* alt çizgeleri belirlemektedir.

Yerellik teoremine göre, bir düğümün ait olduğu *k-çekirdeklilik* değerini hesaplayabilmesi için komşularının çekirdeklilik değerlerini bilmesi yeterlidir. Bu yaklaşımla geliştirdikleri algoritma aşağıdaki gibidir: Başlangıçta her düğüm kendi *k-çekirdeklilik* değeri hakkında bir tahminde bulunur ve bunu komşularına bildirir. Komşularının tahmini değerlerini alan bir düğüm, bu bilgiler doğrultusunda kendi *k-çekirdeklilik* değerini tekrar tahmin eder ve tekrar komşularına gönderir. Bu süreç istenilen bir yakınsama değerine ulaşana kadar sürdürülür. Algoritmalarının sonlanması birkaç yolla olabilmektedir. Bunlar: *i.* Belirli bir süre *k-çekirdeklilik* değeri değişmeyen düğümler çalışmalarını sonlandırabilir, *ii.* Bir önceki roundda hiçbir düğüm yeni tahminde bulunmadığında ya da *iii.* sabit bir round değerine ulaşıldığında sonlanabilir. Yazarlar, gerçek dünya verisi üzerinde yaptıkları testlerin pek çoğunun birkaç 10 roundda sona erdiğini belirtmiştir. Bu nedenle 15-20 arası bir round değerinin, kabul edilebilir sonuç üretmek için yeterli olduğunu açıklamışlardır. Aynı şartlar altında, her zaman en uygun ve hep aynı sonuçları ürettiği için kararlı ve güvenilir bir algoritmadır. Algoritmalarının zaman karmaşıklığının $O(n)$ round ve mesaj karmaşıklığının $O(\Delta m)$ olduğunu açıklamışlardır. Düşük karmaşıklık değerinden dolayı karmaşık ağ uygulamalarında kullanılabilir ve ölçeklenebilir bir algoritmadır. Ama dinamiklik özelliğine sahip değildir.

Sınıf Değerlendirmesi: Bilinirlik oranı, düğümler arası bağlantı sayısıyla ilgilidir. Her düğüm eğer komşularını biliyorsa, kolaylıkla kendi derece merkezilik değerini (C_D) bulabilir. Bu nedenle C_D belirlemek dağıtık uygulamalar için oldukça basit bir yaklaşımdır (Freeman 1977). Basit derece merkezilik algoritmaları yerel veri kullanır ve ucuz maliyetlerle en uygun sonuçları üretir. Anonim derece merkezilik algoritması, bu grubun en iyi bilinen örneklerinden biridir. Bu nedenle pek çok çalışmada kullanılmaktadır ama bazı karmaşık analiz çalışmaları için sadece düğüm derecesine ait veriyi kullanmak yeterli olmayabilmektedir (Kermarrec et al. 2011). Derece merkezilik hesaplamada

kullanılan diğer bir yaklaşım ise diğer düğümlerin derecelerini de kullanmaktır. $C_{DCD}(v)$ (Distributed Cumulative Degree Centrality) ve *PCC* (Decentralized Principal Component Centrality) algoritmaları bu gruba aittir. Bu algoritmalar, yoğun matematiksel işlemler (İng. spectral) yaparak özvektör merkezilikte olduğu gibi bir düğümün merkezilik değerinin hesaplanmasında diğer düğümlerin derecelerini kullanmaktadır. Dağıtık, bilinirlik tabanlı merkezilik algoritmalarının pek çok uygulama alanı vardır. Özellikle, *WSN* uygulamalarında lider belirleme veya küme başı belirlemek (İng. cluster head) en iyi bilinen örneklerdir (Basagni 1999). Ayrıca özvektör benzeri algoritmalar çizgelerde küme çekirdeklerini belirlemek ya da sunucu (İng. server) bilgisayar konumlarını belirlemek için kullanılmaktadır.

Bilinirlik tabanlı merkeziliklerde bir diğer yaklaşım ise *k-çekirdek* ayrıştırma algoritmasıdır. Bu algoritmanın ürettiği sonuçlar bir kenar ya da düğüme ait değildir, sonuçlar mümkün olan en yüksek oranda indirgenmiş alt çizgeleri belirlemektedir. Bu tür algoritmalar, sosyal ağların karakteristik özelliklerini belirlemek, karmaşık ağların ilingesel çizimlerini (İng. visualization of graphs) yapabilmek ve çizgelerde desen belirlemek (İng. motif detection) gibi pek çok alanda uygulanabilmektedir.

Bu sınıfa ait dağıtık algoritmaların zaman ve mesaj karmaşıklıkları genellikle çok düşüktür. Bu nedenle bu algoritmalar ölçeklenebilir, dinamik ve karmaşık ağlarda uygulanabilir algoritmalardır. Fakat hiçbiri dinamiklik (İng. dynamicity) özelliği dikkate alınarak tasarlanmamıştır. Algoritmalar, kullandıkları parametrelere bağlı olarak güvenilir sonuçlar üretebilmektedir.

Çizelge 2.1 Bilinirlik sınıfı algoritmaların özet değerlendirme çizelgesi.

Sınıflar	Merkezilikler	Algoritmalar	Zaman Karmaşıklığı	Mesaj Karmaşıklığı	Ölçeklene-bilirlik	Dinamiklik	Güvenilirlik
Bilinirlik	Derece Merkezilik	Central Degree centrality Algorithm	$O(n)$	_	Yüksek	Hayır	Düşük
		Anonymous distributed degree centrality algorithm	$O(1)$	$O(n\Delta)$	Yüksek	Hayır	Düşük
		Distributed cumulative degree centrality algorithm *(CDCD(v))*	$O(D_G)$	$O(n\Delta D_G)$	Yüksek	Hayır	Katman değ. bağlı
	Özvektör Merkezilik	Central Eigenvector centrality algorithm	$O(n^3)$	_	Düşük	Hayır	Yüksek
		PCC Algorithm	$O(\tau_{mix}\ logn)$	$O(in\Delta)$	Düşük	Hayır	i değ. bağlı
	k-Çekirdek Merkezilik	Central k-Core centrality algorithm	$O(m)$	_	Düşük	Hayır	Yüksek
		Distributed k-core decomposition algorithm	$O(n)$	$O(\Delta m)$	Yüksek	Hayır	Yüksek

2.1.2.2 Erişilebilirlik sınıfı merkezilik algoritmaları

Yakınlık Merkezilik (İng. Closeness Centrality) ($C_C(V)$): Yakınlık merkezilik ölçütü, bir düğümün diğerlerine ne kadar yakın olduğunu ya da çizgedeki ilingesel yoğunluğun merkezine göre konumunu belirtmektedir. En hızlı veri transferinin kısa yollar üzerinden yapılabileceği düşüncesi ile bu yaklaşım kısa yollar üzerine kurulmuş ve tüm iletişimin sadece kısa yollar üzerinden yapıldığı kabul edilmiştir. Bu merkezilik türü sadece düğümler için sonuç üretmektedir. Bildiğimiz kadarı ile ilk tanımlamalar 1950 yılında Bavelas ve 1966 yılında Sabidussi (Sabidussi 1966) tarafından yapılmıştır. Ama günümüz formuna Freeman tarafından getirilmiştir (Freeman 1979).

Çizge $G(V,E)$ iken, $C_C(v)$, v düğümüne ait yakınlık merkezilik değeri, v nin diğer tüm düğümlere olan uzaklıklarının toplamı olarak tanımlanmıştır. Uzaklık tabanlı bir yaklaşım olup aşağıdaki denklem ile hesaplanmaktadır:

$$C_C(v) = \sum_{t \in V} d(v,t) \qquad (2.5)$$

Bir çizgedeki düğümlerin, yakınlık merkezilik değerlerini hesaplamak için normal şartlarda her düğümden her düğüme olan kısa yolların (İng. All Pair Shortest Path *APSP)* hesaplanması gereklidir. *APSP* hesaplamanın yollarından biri de, önce enine arama algoritmasını (İng. Breadth First Search *BFS)* kullanmaktır. *BFS* kullanan, merkezi bir algoritma ile bir düğüme ait yakınlık merkezilik değeri

hesaplamanın zaman karmaşıklığı $O(n^2)$ dir. Tüm düğümler için hesaplanırsa bu maliyet $O(n^3)$ olacaktır (Eppstein and Wang 2001; Lehmann and Kaufmann 2003; Newman 2005; Wehmuth et al. 2011a; Nomikos George 2013; Erciyes 2014). Bu algoritma ölçülebilirlik sınırlarını zorlamakla birlikte, dinamiklik özelliği olmayan bir çözümdür. Güvenilir ve kararlı bir çözümdür. Çünkü tüm çizge verisini kullanarak en iyi sonucu bulmakta ve çizge değişmediği sürece hep aynı sonucu üretmektedir.

Dağıtık Ağ Merkezilik Değerlendirme (İng. Distributed Assessment Of Network Centrality *DANCE*) ve Dağıtık Ağ Yakınlık Merkezilik Değerlendirme (İng. Distributed Assessment Of The Closeness Centrality Ranking *DACCER*) (Wehmuth and Ziviani 2011b; Wehmuth 2012a; Wehmuth and Ziviani 2012b): Wehmuth ve arkadaşları, *DANCE* ve *DACCER* algoritmalarını sunmuşlardır. Birbirine çok benzeyen bu iki algoritmanın ikincisine sadece sıralama bölümü eklenmiştir. Bu çalışma ile Noori'nin çalışması (Noori 2011), birbirlerine çok benzemektedir ve aynı merkezilik hesaplama alt yapısı kullanılmıştır. Her iki çalışma da bir tür dağıtık yığılmalı yakınlık merkezilik yaklaşımı olarak değerlendirilebilir. Noori, "*l-layer* ", Wehmuth ve arkadaşları ise "*h-neighborhood*" adını verdikleri birer komşuluk parametresi kullanmışlardır. Bu *l* ve *h* parametreleri, belli bir düğümden diğerlerine olan uzaklığı veya yerel veri kümesinin çapını temsil etmektedir. Her düğüme ait yerel veri kümesi bu parametreler sayesinde belirlenmektedir. Noori çalışmasında patika uzunluklarını dikkate alarak hesaplama yaparken, Wehmuth ve arkadaşları ise kısa yolların oluşturduğu uzaklıkları kullanarak hesaplama yapmışlardır.

Wehmuth ve arkadaşlarının algoritması, hacim tabanlı bir merkezilik ölçütüdür ve *h-komşuluk* parametresi ile herhangi bir düğüme ait olan yerel veri kümesi (v) belirlenmektedir. Bu küme, *n* gibi bir düğümden *h* kadar uzaklık içinde bulunan düğümleri içermektedir. *v* kümesinin değeri ise küme içindeki düğümlerin derecelerinin toplamı olarak tanımlanmıştır. Örneğin *h=0* ken, *n* düğümüne ait kümenin değeri *v(n),* *n* in derecesine eşittir. *h=1* ken, *n* düğümüne ait kümenin değeri *v(n),* *n* ve *n* in komşularının derecelerinin toplamına eşittir. Bu işlem aşağıdaki denklem ile tanımlanmıştır (Wehmuth et al. 2011a).

$$v(h\_neighborhood(n)) = \sum_{i \in h\_neighborhood(n)} d_i \qquad (2.6)$$

Burada d_i, *n* düğümüne ait dereceyi temsil etmektedir.

Bu çalışma yerel çizge verisi üzerinde çalışan bir yakınsama (İng. approximation) algoritmasıdır. Küme değeri olan $v(h\_neighborhood(n))$ 'yi hesaplama işlemi, her düğümün komşularına kendi derecesi ve mesaj ömrünü (*TTL*) içeren bir mesaj göndermesi ile başlamaktadır. *TTL* değeri *h* parametresine eşitlenerek, mesajların en fazla *h* kadar uzaklığa yayılması sağlanmıştır. Mesaj alan her düğüm, kendi küme değerini güncellemekte ve *TTL* aşılmadığı sürece komşularına iletmektedir. Tüm düğümler bu işlemi aynı anda yaptığı için *h* adım sonra hepsi kendi *h_komşuluk* değerini öğrenmiş olmaktadır.

Bir düğüme ait küme değeri, o düğümden belli bir uzaklıkta olan düğümlere ulaşılarak elde edildiği için yazar, *DANCE* algoritmasının klasik yakınlık merkezilik değeri ile yakından ilişkili olduğunu düşünmektedir. Bu durumu bir deneysel çalışma ile de doğrulamışlardır. Bizim bildiğimiz kadarı ile bu yaklaşım özvektor merkezilik ile de ilişkilidir ve bu ilişki, *h* parametresi çizge çapına yaklaştıkça artacak, tersinde ise azalacaktır.

Düğümlerin aynı anda işlem yapmalarından dolayı, algoritmanın zaman karmaşıklığı, çizge boyutundan bağımsızı olarak *O(h)* olacaktır (Wehmuth et al. 2011a). En kötü durumda, *h* çizge çapına eşit olabilir. Bu durumda zaman karmaşıklığı $O(D_G)$ olacaktır. Bu nedenle ölçeklenebilir yaklaşımdır. Fakat algoritmada dinamiklik özelliği bulunmamaktadır. Yazarlara göre, algoritmalarının mesaj karmaşıklığı $O(n\Delta^h)$ dir. Burada Δ, çizgedeki en büyük düğüm derecesini temsil etmektedir. Bu karmaşıklık, ilk bakışta çok yüksek görünebilir ama yazarların açıklamalarına göre *h=2* gibi bir değere sahipken, algoritmaları kabul edilebilir sonuçlar üretmektedir. Bu nedenle bu maliyetin kabul edilebilir bir düzeyde olduğunu belirtmektedirler (Wehmuth et al. 2011a). Bildiğimiz kadarı ile çizge yapısına ve boyutuna bağlı olarak, *h* ın alacağı küçük değerler için bu algoritmanın en uygun sonucu üretmeme ihtimali vardır.

Yazarlar, son olarak çalışmalarını *Second Order Centrality algorithm SOC* adındaki benzer bir çalışma ile aynı veri kümesini kullanarak kıyaslamışlardır. *SOC*'dan daha hızlı, kararlı sonuç üreten ve daha düşük mesaj maliyetine sahip olan algoritmalarının *SOC* ile benzer sonuçlar ürettiğini gözlemlemişlerdir.

Ad-Hoc Ağlarda, Ağaç Tabanlı Merkezilik Kullanarak Lider Belirleme (İng. Leader Election On Tree-Based Centrality In Ad Hoc Networks *LETBC*) (ChongGun and Mary 2013): Kim ve Wu, hareketli düğümlerin bulunduğu ağlarda (İng. mobile networks), lider belirlemek için kullanılmak üzere

ağaç tabanlı bir merkezilik algoritması sunmuşlardır. Yönlendirme ve teknik bir takım problemlerin denetlenmesi gibi birçok görevi olan lider düğümlerin temel amacı, ağ kaynaklarının etkin kullanımı ve ağın yaşam süresini arttırmaktır. Bu nedenle lider seçiminde düğüm pil düzeyi, diğer düğümlere olan uzaklığı ve işlem kapasitesi gibi temel kısıtlar dikkate alınmaktadır (Vasudevan et al. 2004). Örneğin, kablosuz duyarga ağlarında (İng. Wireless Sensor Networks *WSN)*, eğer lider düğüm yüksek ilingesel merkezilik değerine sahipse büyük miktarda enerji tasarrufu ve veri transferinde hız artışı sağlayabilmektedir.

Bu çalışmada ad-hoc ağlarda kullanılmak üzere, merkezilik tabanlı yeni bir lider seçim yöntemi önerilmiştir. Uzaklık tabanlı bir yaklaşımdır ve bir düğümün coğrafi merkezilik değeri bir ağaç yapısı yardımıyla belirlenmektedir. Merkezilik değeri ise düğümlerin ağaç yapısı üzerindeki ortalama derinlikleri hesaplanarak bulunmaktadır. Bu yaklaşım, sosyal ağlardaki analiz (İng. Social Network Analysis *SNA)* çalışmaları ile kıyaslandığında oldukça basit ve küçük işlemler gerektirmektedir. Çalışma zaman uyumsuz olarak çalışan, yerel çizge verisi kullanan ve dağıtık işlem mimarisine sahip bir yakınsama algoritmasıdır.

Basitçe algoritmanın çalışma mantığı şöyledir: Eğer ağda bir lider yok ise başlatıcı düğüm, komşularına gönderdiği mesajlar ile bir lider seçim işlemi başlatır. Bu mesaj, taşırma yolu ile çizge üzerinde yayılır. Taşırma mesajları yaprak düğümlere ulaştığında ise ebeveynler (İng. parent) yardımıyla aktarılan bilgilendirme mesajları başlatıcı düğüme iletilir. Bu mesajların içinde düğümlere ait derinlik ve seviye verileri bulunmaktadır. İlk amaç, bir dolaşım ağacı (İng. spanning tree) oluşturmaktır. Düğümler, aldıkları mesajlar sayesinde başlatıcı düğüme olan uzaklılarını (derinlik değerlerini) ve ağaçtaki seviyelerini (yaprak düğüme kaç atlama uzaklıkta olduklarını) hesaplayabilmektedir. Örneğin, başlatıcı düğümün derinlik değeri sıfırdır ve ağaçtaki seviyesi ise dolaşım ağacının çapına eşittir. Bir yaprak düğümün derinlik değeri, başlatıcı düğüme olan uzaklığına ve ağaç seviyesi ise sıfıra eşittir. Ağaç oluşumun sonunda, başlatıcı düğüm tüm düğümlere ait derinlik ve seviye bilgilerini elde etmiş olacaktır. Bu verileri kullanarak aşağıdaki denklem yardımıyla, ağaç tabanlı basit merkezilik değeri hesaplanmaktadır.

$$C = \frac{\sum_{i=1}^{n} D_i}{n-1} \qquad (2.7)$$

Burada n çizgedeki düğüm sayısını D_i de düğümlerin derinliğini temsil etmektedir.

Hesaplanan *C* değeri, tam sayıya yuvarlanmakta ve ağaç seviyesi bu *C* değerine eşit olan düğümler birer lider adayı olarak seçilmektedir. Bunların arasından en uygun liderin belirlenmesi ise kalan pil ömrü, düğüm derecesi vb. gibi diğer kısıtlar dikkate alınarak yapılmaktadır.

Yazarların iddiasına göre çalışmaları derece, yakınlık ve aradalılık merkezilik ölçütleri ile benzer sonuçlar üretmekte ve bunu daha düşük mesaj karmaşıklığı ile yapmaktadır. Çalışmada zaman karmaşıklığına yönelik bir bilgi bulunamamıştır. Bildiğimiz kadarı ile bu algoritmanın zaman karmaşıklığı $O(2D_G)$ olmalıdır. Dolaşım ağacı tabanlı bu yaklaşımın, kablosuz ağlar için mesaj karmaşıklığı ise $O(2n)$ olarak verilmiştir. Bizim bildiğimiz kadarı ile bu yaklaşımın kablolu ağlardaki mesaj karmaşıklığı $O(m+n\ D_G)$ olmalıdır. Burada D_G çizge çapını temsil etmektedir.

Yazarlar, düğüm sayısı 30 un altında olan bazı deneysel çalışmalar yapmışlardır. Bu rakam, karmaşık ağlar düşünüldüğünde oldukça küçüktür. Bir kaynaktan başlatılan bir dolaşım ağacı oluşturdukları için sabit bir başlatıcı düğüm önceden bilinmelidir. Ayrıca oluşturulan ağaç, *BFS* ağacında da olduğu gibi bir dolaşım ağacıdır ama en küçük dolaşım ağacı değildir. Tüm düğümler arası kısa yollar dikkate alınmadan yapılan bir çalışma olduğu için büyük ve karmaşık çizgeler için elde edilen sonuçlar güvenilir olmayabilir.

Bu çalışma düşük zaman ve mesaj karmaşıklığı nedeniyle ölçeklenebilirdir. Fakat algoritmanın dinamiklik özelliği yoktur. Ayrıca algoritmada kullanılan küçük mesaj geciktirme süreleri (İng. delay), aynı şartlarda farklı dolaşım ağaçları oluşturulmasına neden olabileceği için sonuçlar güvenilir olmayabilir.

Hızlı Yakınsayan Merkez Bulma (İng. Fast Approximate Center Exploration *FACE*) (Xing et al. 2010): Xing ve arkadaşlarının esas amacı, *WSN*'lerde merkezilik ölçütlerini kullanarak iletişimi yönlendirmek ve özellikle mesaj iletimini hızlandırmaktır. Bu amaçla *FACE* adını verdikleri algoritmaları, çizgedeki en uygun geçiş kapısı (İng. gateway) konumlarını belirlemektedir. Yerel çizge verisi kullanan ve zaman uyumsuz çalışan bir algoritmadır. Çalışmalarında, yerel veri üzerinden yakınlık merkezilik hesaplaması yaparak sonuç üretmişlerdir. Çünkü hedefleri çok hassas ve çizge bazında sonuçlar üretmektir değildir, göreceli olarak sıralayabilecekleri değerler üretmektir. Ayrıca sadece yaprak (İng. leaf) düğümlere ait dolaşım ağaçları kullanarak, kullanılan ağaç sayısını indirgemişlerdir. Bu yolla, işlem maliyetleri azaltılmış ama elde edilen sonuç bir

yakınsama olmuştur. Yazarlara göre, düğümlerin uzaklık tabanlı merkezilik değerleri ile çizge ilingesinin kenarındaki düğümlerin uzaklıkları arasında yakın bir ilişki vardır.

Bu çalışmanın merkezilikle ilgili kısmı şu şekildedir: Merkezilik değeri hesaplanırken her düğüm yaprak düğümlere olan uzaklığını ve örnek nokta (İng. sample point) adı verilen sıradan bir düğüme olan uzaklığını dikkate alır. İlkolarak, sıradan bir düğüm, birincil dolaşım ağacı (İng. Primary Spanning Tree *PST)* adı verilen bir dolaşım ağacı oluşturur. Bu ağaç, yaprak düğümleri belirlemek ve düğümler arası iletişimi gerçekleştirmek için kullanılmaktadır. Daha sonra, her bir yaprak düğüm kendi dolaşım ağacının oluşturur. Bu ağaçlara ikincil dolaşım ağacı (İng. Secondary Spanning Tree *SST)* adı verilmiştir. Bu işlemin sonunda, her düğüm örnek noktaya ve yaprak düğümlere olan uzaklıklarını öğrenmiş durumdadır. Bu uzaklıkları kullanarak, her düğüm merkezilik değerini hesaplar ve *PST* ebeveynine gönderir. Ebeveyn, çocuklarına ait en büyük merkezilik değerini bulur ve kendi *PST* ebeveynine gönderir. Bu sürecin sonunda, örnek noktada (*PST* 'nin kökünde) merkezilik değerleri toplanır bunların içinden en büyüğü seçilir.

Gereken dolaşım ağacı sayısı aşağıdaki denklem ile hesaplanmaktadır;

$$Dolaşım\ ağacı,\ SP = \frac{(n-1)(l+1)}{n} \qquad (2.8)$$

Burada *n* çizgedeki düğüm sayısını ve *l* de *PST* üzerindeki bulunan yaprak düğüm sayısını temsil etmektedir.

Yukarıdaki denklemden de açıkça anlaşılacağı gibi, *n* nin büyük değerleri için *SP = (l+1)* olacaktır. *FACE* algoritmasının zaman karmaşıklığı $O(D_G)$ olarak verilmiştir. Burada D_G, çizge çapını temsil etmektedir. Düşük zaman karmaşıklığından dolayı bu algoritma ölçeklenebilirdir. Fakat algoritmanın dinamiklik özelliği yoktur. Çalışmada mesaj karmaşıklığına yönelik bir açıklamaya yer verilmemiştir. Hesaplamalarımıza göre, bu algoritmanın mesaj karmaşıklığı *O(2mSP)* dir.

Karmaşık Ağların Değerlendirilmesi İçin Merkezi Olmayan Algoritmalar (İng. Decentralized Algorithms For Evaluating Centrality In Complex Networks) (Lehmann and Kaufmann 2003): Lehmann ve Kaufmann yerel veri kullanarak elde ettiği kısa yolları kullanan, dağıtık olarak çalışan ve bazı

düğüm merkezilik hesaplamalarına uygun genel bir alt yapı sunmuşlardır. Bu çalışmada sadece komşuluk bilgisi ve çizge çapı verisi kullanılmasına rağmen elde edilen sonuçlar standart merkezilik algoritmalarının sonuçları ile yakın benzerlik göstermiştir. Diğer bir ifade ile yerel veriler kullanılarak, genel sonuçlar elde edilmiştir. Bu nedenle algoritmalarının güvenilirliği yüksektir. Genel olarak bakıldığında bu alt yapı, zaman uyumlu çalışan bir tür *APSP* algoritmasıdır.

Bu yaklaşım iki bölümden oluşmaktadır. Birinci bölüm kısa yolların belirlenmesi, ikinci bölüm ise raporlama aşamasıdır. Birinci bölüm, merkezilik algoritmaların ortak kullandığı bölümdür. Yakınlık, gerginlik ve aradalılık merkezilik hesaplamaları için gereken kısa yollar belirlenir. Bu amaçla her düğüm, komşularına taşırma yolu ile mesajlar gönderir. Bu bölüm, bir tür zaman uyumlu *BFS* ağaç algoritması gibi çalışır. İkinci bölüm, ilk bölüm tamamlandıktan sonra düğümlerin bilgilerini paylaştığı bölümdür. Bu bölümde merkezilik türüne göre yapılan işlemler farklılıklar göstermektedir. Bu yolla her düğümün kendi merkezilik değerini hesaplaması sağlanmıştır.

Algoritmanın sonlanması, çizge çapı kontrolü ile yapılmaktadır. Mesaj yayılımı çizge çapına eşit olduğu için algoritmaların zaman karmaşıklığı $O(D_G)$ dir. D_G burada çizge çapını temsil etmektedir. Bu nedenle çalışmada önerilen algoritmalar, yüksek ölçüde ölçeklenebilirdir ve karmaşık ağlara kolaylıkla uygulanabilir. Fakat bu çalışmada dinamiklik özelliği dikkate alınmamıştır. Algoritmaların mesaj karmaşıklığı ise $O(n^2m)$ dir. Yönlü ve yönsüz çizgelerde çalışabilen bu yaklaşım, karmaşık ağlar düşünüldüğünde, oldukça büyük mesaj karmaşıklığına sahiptir (Wehmuth and Ziviani 2011b) ve bu durum ölçeklenebilirliğin sınırlarını zorlayacaktır. Bu çalışma bir teknik rapor olarak yayınlanmış ve bildiğimiz kadarı ile daha sonra bu konu üzerine yazarlarca bir çalışma yapılmamıştır.

Sınıf Değerlendirmesi: Erişilebilirlik değeri düğümler arası uzaklıklara bağlıdır. Esas amaç, çizgedeki tüm düğümlere en yakın düğümü bulmaktır. Yakınlık merkezilik ölçütü bu amaçlar için bilinen en uygun yöntemlerden biridir. Klasik yakınlık merkezilik yaklaşımı uzaklık tabanlıdır ve hesaplanabilmesi için *APSP* gerektirir. Merkezi bir *APSP* algoritmasının zaman karmaşıklığı $O(n^3)$ dir. Bu değer, karmaşık ağlar düşünüldüğünde ölçeklenebilirliğin üst sınırlarındadır. Bu karmaşıklık değerini düşürebilmek için *DANCE/DACCER (Wehmuth 2012a; Wehmuth and Ziviani 2012b)* gibi algoritmalar, sınırlı komşuluk parametreleri kullanarak küme değerleri hesaplamışlardır. *LETBC* (ChongGun and Mary 2013)

gibi algoritmalar, sadece bir dolaşım ağacı kullanmışlar ve *FACE* (Xing et al. 2010) gibi algoritmalar ise sınırlı sayıda dolaşım ağacı kullanarak merkezilik hesaplamışlardır.

Bu sınıfta incelenen algoritmaların hepsi yerel veri kullanmaktadır ve zaman karmaşıklıkları çizge çapına eşittir. Algoritmaların güvenilirlikleri ve mesaj karmaşıklıkları ise kullanılan bazı parametrelere göre (yerel veri miktarı, kullanılan dolaşım ağacı sayısı vb.) değişiklik göstermektedir. Bu nedenle *LETBC* gibi bazı algoritmaların, çizge verisinin çok azını kullanarak merkezilik değeri hesaplamalarından dolayı güvenilirlikleri düşüktür ve genel bir merkezilik ölçütü olarak değerlendirilmeleri yanlış olacaktır. Bu tür çalışmalar özel amaçlı yaklaşımlar olarak değerlendirilmelidir. *DANCE/DACCER* algoritmasının güvenilirliği ise seçilen parametrelere bağlı olarak özvektör ile derece merkezilik arasındadır. *FACE* ise karmaşık ağ uygulamalarına yukarıdakilerden daha uygundur. Kullanılan ağaç sayısı yaklaşık olarak yaprak sayısına eşittir. Ayrıca üretilen sonuçların hassasiyeti de öncekilerden daha yüksektir. Ama kullanılan dolaşım ağaçları, en küçük dolaşım ağacı olmadığı için kısa yollar dışındaki yolları da kullanabilecektir. Bu da sonuçların güvenilirliğini ve en iyiliğini zayıflatacaktır. Lehman ve arkadaşlarına ait olan genel amaçlı algoritma ise zaman uyumlu olarak çalıştığı ve *APSP* hesabı yaptığı için mesaj karmaşıklığı diğerlerinden daha yüksektir ama kararlı ve en iyiye yakın değerler üreten bir algoritmadır. Zaman uyumlu olması ve yüksek mesaj maliyeti, karmaşık ağlara uygulanmasını zorlaştırmasına rağmen bu sınıfın en iyilerinden biridir.

Sonuç olarak bu sınıftaki çalışmalar, genellikle çizgedeki diğer düğümlere en yakın olan düğümü bulmak için kullanılır ve düşük zaman karmaşıklarından dolayı ölçeklenebilirdirler. Fakat hiçbirinin tasarımında dinamiklik özelliği dikkate alınmamıştır.

Çizelge 2.2 Erişilebilirlik sınıfı algoritmaların özet değerlendirme çizelgesi.

Sınıflar	Merkezilikler	Algoritmalar	Zaman Karmaşıklığı	Mesaj Karmaşıklığı	Ölçeklenebilirlik	Dinamiklik	Güvenilirlik
Erişilebilirlik	Yakınlık Merkezilik	Central Closeness centrality algorithm	$O(n^3)$	_	Düşük	Hayır	Yüksek
		DANCE and DACCER Algorithms	$O(D_G)$	$O(n\Delta^h)$	Yüksek	Hayır	*h* değerine bağlı
		LETBC Algorithm	$O(D_G)$	$O(m+nD_G)$	Yüksek	Hayır	Çok düşük
		Decentralized algorithms for evaluating centrality in complex networks	$O(D_G)$	$O(n^2m)$	Yüksek	Hayır	Yüksek
		FACE Algorithm	$O(D_G)$	$O(mSP)$	Yüksek	Hayır	*SP değ. bağlı*

2.1.2.3 Geçirgenlik sınıfı merkezilik algoritmaları

Bu sınıfta, bulunabilen çalışmalar üç temel merkezilik yaklaşımı dikkate alınarak incelenmiştir.

Gerginlik Merkezilik (İng. Stress Centrality) ($C_S(V)$): Bu merkezilik türüne göre bir düğüm, üzerinden geçen kısa yol sayısı kadar önemlidir ve bu sayı ilgili düğümün gerginlik merkezilik değerini vermektedir. Uzaklık tabanlı bir yaklaşımdır ve tüm iletişimin sadece en kısa yollar üzerinden yapıldığı kabul edilmektedir. Bildiğimiz kadarı ile ilk defa 1953 yılında Shimbel tarafından geliştirilmiştir (Freeman 1977; Freeman 1979; Brandes 2001).

Çizge $G(V,E)$, $s,v,t \in V$ ve $s \neq t \neq v$ *iken,* v gibi bir düğüme ait gerginlik merkezilik değeri $C_S(v)$, her s ve t ikilisi için v üzerinden geçen kısa yolların toplamına eşittir. Bu ifade aşağıdaki denklemde gösterilmiştir.

$$C_S(v) = \sum_{s \in V} \sum_{\substack{t \in V \\ s \neq t \neq v}} \sigma_{st}(v) \qquad (2.9)$$

$\sigma_{st}(v)$ yi hesaplamak için bir tür *APSP* algoritması çalıştırılmalıdır. Bu amaçla *BFS* algoritması düşünüldüğünde, merkezi bir gerginlik merkezilik algoritmasının zaman karmaşıklığı $O(n^3)$ olacaktır (Lehmann and Kaufmann 2003; Erciyes 2014). Bu karmaşıklık değeri ile ölçeklenebilirlik sınırları zorlanmakla birlikte, algoritma tüm çizge verisi üzerinden sonuç ürettiği için güvenilir bir yaklaşımdır. Algoritmada dinamiklik özelliği bulunmamaktadır.

Kısıtlanmış Gerginlik Merkezilik (İng. Restricted Stress Centrality) Algoritması (P. Fekete et al. 2005): Fekete ve arkadaşları, bir duyarga ağındaki algılayıcıların fiziksel yerleşim alanın sınırlarını belirleyebilmek amacıyla bu çalışmayı yapmışlardır. Elde edilen sınır bilgisi ile bölgeye yapılan giriş çıkışların izlenebilmesi, alan denetimi gibi değişik birçok amaçla kullanılabilmektedir. Çalışmaları gerginlik merkezilik tabanlı, dağıtık olarak tasarlanmış ve yerel veri kullanan bir yakınsama algoritmasıdır. Kısa yolların hesaplanmasında önceden tanımlanmış δ ile gösterilen bir uzaklık parametresi kullanmışlardır.

Yazarlara göre çalışmaları bilinen pek çok merkezilik türü içinde geometrik ağlar için en uygun olan çalışmadır. Ayrıca bu uygunluğu kanıtlamak için matematiksel ve deneysel birtakım kanıtlar sunmuşlardır. Denklem 2.9 da tanımlanan standart gerginlik merkezilik eşitliğine, uzaklık parametresi olan *δ'i* ilave ederek aşağıdaki eşitliği tanımlamışlardır. Böylece yalnız istenen uzaklıktaki düğümlerin dikkate alınması sağlanmıştır.

$$Rstress(v,\delta) = \sum_{\substack{s,t\in V_\delta(v) \\ s\neq t\neq v}} \sigma_{st}(v) \qquad (2.10)$$

Burada $V_\delta(v)$, *v* düğümünden δ uzaklıktaki düğümlerin kümesidir.

Rstress(v,δ) merkezilik değerinin hesaplanmasına yönelik açıklamalar oldukça sınırlıdır. Sınırlı açıklamalar doğrultusunda, anladığımız kadarı ile algoritmanın temel adımları aşağıdaki gibidir:

Öncelikle her düğüm, kendisine δ uzaklıktaki düğümler arasındaki kısa yolları belirlemektedir. Bunu yaparken düğümler komşularına yaşam ömrü (*TTL*) δ ' ya eşit olan mesajlar göndermektedir. Bu mesajı ilgili düğümden ilk defa alan her düğüm kendi $\sigma_{st}(v)$ değerini arttırmakta ve mesajın ömrü dolmadı ise komşularına iletmektedir. Bu işlemin sonunda tüm düğümler kendi *Rstress(v,δ)* değerini hesaplamış olmaktadır. Daha sonra sınırdaki düğümleri belirlemek için önceden tanımlanan bir eşik değeri (İng. threshold) kullanılmaktadır. Bu değerden küçük merkezilik değerine sahip olanlar bölgenin sınırındaki düğümler, diğerleri ise bölgenin içindeki düğümler olarak kabul edilmektedir.

Bu çalışmanın, ağdaki düğümlerin yerleşimi ve yoğunluğu üzerine getirdiği sınırlamalar önemli zayıflıklarındandır. Gerçek hayatta, duyarga ağları daha seyrek ve rastgele bir yoğunlukta yerleşmiş olabilir (Wei et al. 2012). Ayrıca ağın sadece dış sınırlarını belirlemek amacıyla tasarlanmış olan bu çalışmanın,

karmaşık ağların analizinde kullanılıp kullanılamayacağına yönelik bir bilgiye de rastlanamamıştır.

Algoritmanın zaman karmaşıklığı *O(δ)* dir. Çünkü her düğüm sadece δ kadar uzaklığa mesaj iletmektedir. Bu nedenle ölçeklenebilirlik oranı oldukça yüksektir. Çalışmanın dinamiklik özelliği bulunmamaktadır. Mesaj karmaşıklığı ise δ mesafede olan kenar sayısı ile ilgilidir. Bu nedenle en kötü durumda, bir düğüm için algoritmanın mesaj karmaşıklığı *O(m)* dir. Tüm düğümler dikkate alındığında ise *O(nm)* olacaktır. Açıkça görülebileceği gibi δ değeri çizge çapına yaklaştıkça, algoritmanın karmaşıklığı ve elde edilen sonuçlar klasik gerginlik merkezilik algoritması değerlerine yakınsayacaktır. Bu nedenle algoritma sonuçlarının güvenilirliği doğrudan δ parametresinin büyüklüğüne bağlıdır.

Karmaşık Ağların Değerlendirilmesi İçin Merkezi Olmayan Algoritmalar (İng. Decentralized Algorithms For Evaluating Centrality In Complex Networks) (Lehmann and Kaufmann 2003): Yakınlık, gerginlik ve aradalılık merkezilik değerleri hesaplayabilen genel amaçlı bir algoritmadır. Bu algoritma hakkında detaylı bilgi bölüm 2.1.2.2 de verilmiştir.

Aradalılık Merkezilik (İng. Betweenness Centrality) ($C_B(V)$): Aradalılık merkezilik *(C_B),* diğerlerinden daha çok kullanım alanı olan ve oldukça karmaşık bir ölçüttür. Gerginlik merkezilik ölçütünün ileri bir sürümü olarak düşünülebilir. Bu türde de tüm iletişimin sadece kısa yollar üzerinden yapıldığı kabul edilmektedir. Yüksek aradalılık merkezilik değerine sahip bir düğüm, üzerinden yoğun trafik geçiren, kesim (İng. cut point), tıkanıklık ya da köprü noktası olarak tanımlanabilir (Green et al. 2012). Bu düğümler, ağ üzerinde yoğun kullanılan ve üzerinden çok fazla kısa yol geçiren düğümlerdir. Bu nedenle çok farklı kullanım alanları vardır. Örneğin ağlarda tıkanıklık noktalarını önceden belirlemek için kullanılabildiği gibi çizge ile modellenebilen sistemlerde (yönlendirici (İng. router) ağları, atıf ağları (İng. co-citation networks), sosyal ağlar vb.) bir analiz aracı olarak da kullanılabilmektedir (Newman 2000; Holme 2003; Newman 2005). Bu merkezilik türünde uzaklık ve trafik verisi birlikte değerlendirilerek sonuç üretilmektedir. Bildiğimiz kadar ile ilk defa 1954 yılında Shaw tarafından tanımlanmıştır. Modern formuna ise Freeman tarafından getirilmiştir (Freeman 1977; Freeman 1979). Temel olarak düğüm ve kenar aradalılık merkezilik şeklinde iki farklı türü bulunmaktadır.

Çizge *G(V,E), s,v,t* $\in$ *V* ve *s≠t≠v* iken, *v* gibi bir düğüme ait düğüm aradalılık merkezilik değeri $C_{BV}(v)$, bu düğümün diğer düğümlerce kullanılma oranlarının toplamı alınarak bulunmaktadır. Bu işlem aşağıdaki denklem ile gösterilmiştir.

$$C_{BV}(v) = \sum_{s\in V} \sum_{\substack{t\in V \\ s\neq t\neq v}} P_{st}(v) = \sum_{s\in V} \sum_{\substack{t\in V \\ s\neq t\neq v}} \frac{\sigma_{st}(v)}{\sigma_{st}} \qquad (2.11)$$

Kenar aradalılık merkezilik hesaplamasında ise düğümün yerine kenar üzerinden hesaplama yapılmaktadır. Aynı şartlarda, *e = (u,v)* ve *u,v* $\in$ *V* iken, *e* gibi bir kenara ait kenar aradalılık merkezilik değeri $C_{BE}(u,v)$, bu kenarın diğer düğümlerce kullanılma oranlarının toplamı alınarak hesaplanmaktadır. Bu işlem aşağıdaki denklem ile gösterilmiştir.

$$C_{BE}(u,v) = \sum_{s\in V} \sum_{\substack{t\in V \\ s\neq t\neq u\neq v}} P_{st}(u,v) = \sum_{s\in V} \sum_{\substack{t\in V \\ s\neq t\neq u\neq v}} \frac{\sigma_{st}(u,v)}{\sigma_{st}} \qquad (2.12)$$

Sonuç olarak, geleneksel aradalık merkezilik hesaplamaları temel olarak iki adımda gerçekleştirilmektedir. Bunlar; *i.* Çizgedeki tüm düğümler arasındaki uzaklıklar belirlenerek olası her ikili arasındaki kısa yolların sayısının bulunması, *ii.* Bir ikilinin kaç tane kısa yolunun ilgili kenar ya da düğüm üzerinden geçtiğinin bulunması ve bu değerlerin toplanması (Lehmann and Kaufmann 2003; Borgatti and Everett 2006; Erciyes 2014), (Barthélemy 2004; Newman 2005; Zhuge and Zhang 2009; Nomikos George 2013).

Yakınlık algoritmasında olduğu gibi burada da merkezi *BFS* ağaç algoritması kullanılarak kısa yollar belirlenebilir. Bu durumda, geleneksel algoritmanın zaman karmaşıklığı $O(n^3)$ olacaktır (Wehmuth et al. 2011a). Fakat ağırlıksız çizgelerde çalışan merkezi Brandes aradalılık algoritması kullanılırsa, zaman karmaşıklığı *O(nm)* olacaktır (Brandes 2001; Newman 2005; Erciyes 2014). Bu değerler ile aradalılık algoritması, ölçeklenebilirliğin sınırlarını zorlamaktadır. Algoritmalar, dinamiklik özelliğine sahip değildir ama tüm çizge verisini kullanan kararlı yapılarından dolayı güvenilir sonuçlar üretmektedir.

Benlik Aradalılık Merkezilik (İng. Ego Betweenness Centrality *EBC*) Algoritması(Guidi et al. 2014): Guidi ve arkadaşları, dağıtık mimaride bir yakınsama algoritması olan *EBC* algoritmasını sunmuşlardır. Bu çalışma yönlü ya da yönsüz alt yapısı olan, dinamik ve dağıtık ağlarda kullanılabilmektedir. Çalışmalarında yerel veri kullanmaktadırlar. Kullanılan veri, ilgili düğümün

çevresindeki belirli sayıda düğüme ait olduğu için bu veriye benlik ağı (İng. ego network) adını vermişlerdir.

Benlik aradalılık merkezilik değerinin hesaplanması için genel mesaj yayılımı (İng. broadcast) tekniği ve dedikodu yayılım (İng. gossip) tekniği şeklinde iki farklı algoritma kullanılmıştır. Bu iki algoritma arasında, sadece komşuluk matrisinin güncellenmesi kısmında farklılık bulunmaktadır. Her düğüm komşuluk matrisini elde ettikten sonra *EBC* değerini matematiksel yollarla hesaplanmaktadır. Yönsüz çizgelere ait *EBC* hesaplaması için gerekli eşitlik Denklem 2.13 da verilmiştir. Bu çalışma dinamiklik özelliğinden dolayı, ağdaki değişikliklerin (düğüm/kenar eklenmesi ya da silinmesi) farkına varmakta ve düğümler arası güncelleme mesajları ile *EBC* değerlerinin ilgili düğümlerde yeniden hesaplanmasını sağlamaktadır.

Genel yayın tekniğini kullanan algoritmada her düğüm, genel yayınlanan mesajlar yardımıyla, birbirlerine ait komşuluk bilgilerini (*EgoΓ(v)*) paylaşmaktadır. Bu nedenle açıklamalardan anladığımız kadarı ile algoritmanın dinamiklik özelliği hariç, bir düğüm için zaman karmaşıklığı *O(1)* ve mesaj karmaşıklığı ise *O(|EgoΓ(v)|)* dir.

Dedikodu yayılım tekniğini kullanan algoritmada, *UpdateRequest* ve *UpdateReply* adını verdikleri iki tür mesaj kullanmışlardır. Bu mesajlar, komşuluk bilgilerinin paylaşımında kullanılmaktadır. Bir düğüm komşuluk matrisini hazırlamak amacıyla, rastgele seçtiği bir komşusuna *UpdateRequest* mesajı gönderir. Bu mesajı alan bir düğüm, komşuluk bilgisi ile birlikte *UpdateReply* mesajı gönderir. *UpdateReply* mesajı alan düğüm *w* kadar bir süre bekledikten sonra tekrar rastgele seçtiği bir komşusuna *UpdateRequest* mesajı gönderir. Bu işlem tüm komşularına *UpdateRequest* mesajı gönderene kadar devam eder. Bu algoritmanın dinamiklik özelliği hariç tutulduğunda, açıklamalardan anladığımız kadarı ile bir düğüm için zaman karmaşıklığı en az *Ω(|EgoΓ(v)|w)* ve mesaj karmaşıklığı ise en az *Ω(|EgoΓ(v)|)* olacaktır.

Yönsüz çizgeler için bu işlem aşağıdaki denklem ile tanımlanmıştır.

$$EBC(n) = \sum_{A_n(i,j)=0, j>i} \frac{1}{A_n^2(i,j)} \qquad (2.13)$$

Burada A_n, *n* gibi bir düğümün benlik ağına ait komşuluk matrisidir. Çalışmada algoritmanın geneline yönelik bir karmaşıklık değeri verilmemiştir.

Diğer çalışmalarla arasındaki farkı daha net ortaya koyabilmek amacıyla karmaşıkları kıyaslanırken bu algoritmaların dinamiklik özelliği hariç tutulmuştur. *EBC* algoritması düşük karmaşıklık değerlerinden dolayı ölçeklenebilirdir ve dinamiklik özelliğine sahiptir. Anladığımız kadarı ile bu çalışmada bir kenar uzaklıktaki komşular dikkate alınarak işlem yapılmıştır. Bu nedenle elde edilen sonuçlar tüm çizge verisini temsil etmeyecektir. Ayrıca yazarlar da, deneysel çalışmalarının sonucuna bakarak, *EBC* ile standart aradalılık merkezilik sonuçları arasındaki benzerliğin (İng. correlation) 100 düğümden büyük çizgelerden sonra kaybolduğunu belirtmişlerdir. Bu nedenle bu yaklaşımlar karmaşık ağ analizinde kullanılabilir ama elde edilen sonuçların güvenilirliği sorgulanmalıdır.

Aktarıcı Aradalılık (İng. Sink Betweenness *SBet)* Algoritması (Oliveira et al. 2010): Oliveira ve arkadaşlarının amacı, merkezilik ölçütlerini kullanarak kablosuz duyarga ağlarında (Wireless Sensor Networks *WSN*) mesaj yönlendirme alt yapısı oluşturmaktır. Bu amaçla dağıtık mimaride yeni bir ilingesel ölçüt olan ve *Sink Betweenness* adını verdikleri çalışmalarını sunmuşlardır. Bu çalışma belli bir kaynak düğüm tarafından yönetilen (İng. single source initiator), kısa yol tabanlı, ağırlıksız çizgeler üzerinde yerel veri ile zaman uyumsuz olarak çalışan bir yakınsama algoritmasıdır.

Bu çalışmanın temeli, merkezilik değeri yüksek düğümlerin birer aktarma düğümü (İng. relay node) gibi kullanılabileceği düşüncesine ve *WSN'* lerdeki veri trafiğinin duyarga düğümlerden aktarıcı (İng. sink) düğümlere doğru olmasına dayanmaktadır. Bu nedenle bu yaklaşımda, sadece başlatıcı/aktarıcı düğümden başlatılan bir tek kısa yol ağacı kullanılmıştır. Geleneksel aradalılık merkezilik ölçütleri hesaplanırken *APSP* kullanıldığı için bu çalışma oldukça sınırlı ölçüde aradalılık hesabı yapan bir çalışmadır. *SBet*, düşünülen anlamda karmaşık ağların analizinde kullanılmayabilir. Fakat yazarların amaçlarına ulaşması için yeterlidir. Çünkü sadece tek yönlü ve sonuçta tüm verinin tek bir düğümde toplanacağı bir yönlendirme alt yapısı oluşturmayı amaçlamışlardır. Bu yaklaşım aşağıdaki denklemde gösterilmiştir.

$$SBet(v) = \sum_{\substack{t \in V \\ s \neq t \neq v}} \frac{\sigma_{st}(v)}{\sigma_{st}} \qquad (2.14)$$

Burada $s \neq t \neq v$ iken, s aktarıcı düğümü, $\sigma_{st}(v)$ ise v üzerinden geçen, t' den s'e olan kısa yol sayısını ve σ_{st} ise t' den s'e olan kısa yol sayısını temsil etmektedir.

Bu çalışmanın merkezilik hesaplanmasına yönelik kısmı aşağıdaki gibidir: İlk olarak başlatıcı düğümden başlayan ve duyarga düğümlere doğru gelişen bir kısa yol ağacı oluşturulur. Bu bir tür dağıtık *BFS* algoritması olarak düşünülebilir. *BFS*' ten farklı olarak alternatif olan tüm kısa yollar belirlenmektedir. Kısa yolları belirlerken, başlatıcı düğümden yaprak düğümlere doğru taşırma yöntemi ile yayılan "hello" mesajı kullanılmıştır. Daha sonra aradalılık hesabı yaparken, yaprak düğümlerden başlatıcı düğüme doğru ilerleyen "border" mesajı kullanılmıştır. Algoritmada kısa yolları yakalayabilmek için t_{delay} adını verdikleri gecikme süreleri kullanılmıştır. Bu nedenle çalışmaları gecikme ağacı (İng. delay tree) olarak da bilinmektedir (Ramos et al. 2014).

Algoritmanın mesaj karmaşıklığı, toplam "Hello" and "Border" mesajlarının sayısına eşit olacaktır. Ağ kablosuz olduğu için her düğüm sadece birer kere bu mesajlardan gönderecektir ve mesaj karmaşıklığı $O(2n)$ olacaktır. Kablolu ağlar için ise mesaj karmaşıklığı $O(2m)$ olacaktır. Algoritmanın zaman karmaşıklığı hakkında bir açıklamaya erişilememiştir. Hesaplamalarımıza göre, mesaj yayılım süresi çizge çapına (D_G) eşit olacaktır. Her atlamada t_{delay} kadar bir gecikme kısa yolları yakalamak amacıyla eklendiği için algoritmanın zaman karmaşıklığı $O(2\ D_G\ t_{delay})$ olmalıdır.

Gecikme zamanının çok küçük olduğu düşünüldüğünde, algoritmanın ölçeklenebilirliği kabul edilebilir sınırlar içindedir ama dinamiklik özelliğine sahip değildir. Kısa yolları yakalamak için kullanılan t_{delay} süresinin seçimi ise sonuçların güvenilirliği açısından oldukça etkilidir. Çünkü bu değere göre, belirlenen kısa yollar değişebilmektedir. Bu nedenle bizim düşündüğümüz anlamda güvenilirliği zayıftır ve bildiğimiz kadarı ile karmaşık ağlarda uygulanabilirliği düşük bir yaklaşımdır.

Yerelleştirilmiş Köprülük Merkezilik (İng. Localized Bridging Centrality *LBC*) Algoritması (Nanda and Kotz 2008): Nanda ve Kotz, düğüm aradalılık merkezilik ölçütünü kullanarak sınıflandırma yapan (İng. clustering) dağıtık bir yakınsama algoritması sunmuşlardır. Kısa yollar üzerine dayanan bu algoritma yerel veri kullanarak çalışmaktadır. Bu çalışma Hwang'a ait merkezi köprü merkezilik algoritmasından türetilmiştir (Hwang et al. 2006).

Bu çalışmada geleneksel düğüm aradalılık merkezilik ölçütü, sadece ilgili düğümün bir atlama (İng. hop) uzaklıktaki komşularından oluşan yerel veri üzerinde uygulanmaktadır (Kermarrec et al. 2011). Bu nedenle bu yaklaşım

Egocentric Betweenness Centrality ($C_{EgoB}(v)$) olarak adlandırılmıştır. Geleneksel aradalılık merkezilik ise tüm çizge verisini kullandığı için *Sociocentric Betweenness Centrality* olarak adlandırılmaktadır. Ayrıca tanımladıkları yerel veri yapısının, dağıtık ve paralel uygulamalar için de uygun bir yaklaşım olduğunu belirtmişlerdir. Bu çalışmada *Bridging Coefficient* (*BC(v)*) adı verilen bir tanım daha kullanılmıştır. Bir *v* düğümüne ait *BC(v)* değeri, o düğümün yerel köprülük ölçütünü yansıtmaktadır. Aşağıdaki denklemde hesaplama şekli gösterilmiştir.

$$BC(v) = \frac{\frac{1}{deg(v)}}{\sum_{i \epsilon \Gamma(v)} \frac{1}{deg(i)}} \qquad (2.15)$$

Burada *Γ(v), v* nin komşu kümesini *deg(v)* ise derecesini temsil etmektedir. Yerelleştirilmiş köprülük merkezilik ölçütü, $(C_{LBr}(v))$ ise aşağıdaki şekilde hesaplanmaktadır:

$$C_{LBr}(v) = BC(v)\ C_{EgoB}(v) \qquad (2.16)$$

Burada $C_{EgoB}(v)$ ise *v*'nin yerel aradalılık merkezilik değeridir ve yerel veri üzerine uygulanan standart aradalılık merkezilik ölçütü $C_{BV}(v)$ ile hesaplanmaktadır. $C_{BV}(v)$'nin hesaplama yöntemi Denklem 2.11 de tanımlanmıştır. Yerel veri sadece ilgili düğümün komşularından oluştuğu için $C_{BV}(v)$ hesaplamasında kullanılan *n* yerine artık çizgedeki en büyük düğüm derecesi *(Δ)* kullanılmalıdır. Yazarlar açıklamalarında yeni bir aradalılık algoritması üretmediklerini, Lehmann ve arkadaşlarına (Lehmann and Kaufmann 2003) ait olan çalışmayı kendi çalışmalarına göre uyarladıklarını belirtmişlerdir.

Bu çalışmada amaç, köprü konumundaki düğümleri bularak sınıflandırma yapmaktır. Bu amaç için tüm düğümlere ait C_{LBr} hesabı yapıldıktan sonra listenin ilk %25 lik diliminin uygun adayları içerdiği kabuledilmiştir. *LBC* yaklaşımı, hesaplama işlemini tekrar gerektirmeden yapmaktadır. Yani çizgedeki köprülerin hepsi standart köprü algoritmalarının tersine, *LBC* algoritmasının bir kere çalıştırılması ile belirlenebilmektedir. Bu nedenle oldukça hızlı bir yaklaşımdır, dağıtık ve paralel uygulamalar için oldukça uygundur. Ayrıca yönlü ve yönsüz çizgelerde çalışabilen bu yaklaşım, kolaylıkla kenar aradalılık merkezilik ölçütüne de çevrilebilmektedir.

Bu çalışma küçük ölçekli kablosuz ağlar için önerilmiş olmasına rağmen, aslında sosyal ağların analizinde kullanılmak üzere tasarlanmıştır. *LBC*'nin esin kaynağı Marsden (Marsden 2002) in çalışmasıdır. Marsden, yerelleştirilmiş ve

genel merkezilik ölçütleri arasındaki ilişkiyi deneysel kanıtlarla göstermiştir (Wehmuth et al. 2011a). Nanda ve Kotz da bu noktaya dikkat çekerek; eğer kullanılan çizge verisinin benlik (İng. egocentric) aradalılık merkezlik ile genel (İng. sosyacentric) aradalılık merkezilik değerleri arasından bir ilişki yoksa bu çalışmanın sonuçlarının güvenilir olmayacağını belirtmişlerdir (Nanda and Kotz 2008). Bir atlama mesafedeki düğümlerden oluşan yerel veri, karmaşık ağlar düşünüldüğüne oldukça sınırlı bir yaklaşımdır. Bu nedenle bazı araştırmacılar, çalışmalarında atlama sayısını değiştirmek yerine uzaklık parametresi kullanmışlardır (P. Fekete et al. 2005; Noori 2011; Wehmuth 2012a). Bir atlama mesafedeki komşuluk ağından dolayı, *LBC* yaklaşımının ürettiği sonuçların hassasiyeti düşüktür ve güvenilirliği sorgulanabilir durumdadır.

Bu algoritmanın karmaşıklıkları ile ilgili bir açıklamaya rastlanmamıştır. Lehman ve arkadaşlarına ait dağıtık algoritmayı (Lehmann and Kaufmann 2003) uyarladıkları için onunla benzer karmaşıklık değerlerine sahip olduğu kabuledilmiştir. Orjinal algoritma tüm çizge verisini kullandığı için karmaşıklıkları yerel veriye göre uyarlarken, *n* yerine en büyük düğüme ait derece *(Δ)* ve çizge çapı yerine de bir değeri kullanılmalıdır. Bu durumda LBC yaklaşımının zaman karmaşıklığı *O(1)* ve mesaj karmaşıklığı ise $O(\Delta n m_{Ego})$ olacaktır. Burada m_{Ego}, *1* atlama uzaklıktaki komşu kümesine ait kenarları temsil etmektedir. Bu algoritma, düşük karmaşıklık değerlerinden dolayı ölçeklenebilirdir ama dinamiklik özelliği yoktur.

Karmaşık Ağların Değerlendirilmesi İçin Merkezi Olmayan Algoritmalar (İng. Decentralized Algorithms For Evaluating Centrality In Complex Networks) (Lehmann and Kaufmann 2003): Yakınlık, gerginlik ve aradalılık merkezilik değerleri hesaplayabilen genel amaçlı bir algoritmadır. Bu algoritma hakkında detaylı bilgi bölüm 2.1.2.2 de verilmiştir.

En İyiye Yakın Dağıtık Aradalılık Merkezilik (İng. Nearly Optimal Distributed Algorithm For Computing Betweenness Centrality NODBC) Algoritması (Hua et al. 2016): Hua ve arkadaşları, doğrusal zamanda *(O(n))* düğüm aradalılık merkezilik ölçütü hesaplayan zaman uyumlu dağıtık bir algoritma sunmuşlardır. Algoritmaları yönsüz ve ağırlıksız çizgeler üzerinde çalışmaktadır ve iletilen mesaj boyu *O(log(n))* bit ile sınırlıdır. Çalışmalarının bu özellikleri taşıyan ilk algoritma olduğunu belirtmişlerdir.

NOBDC algoritması, Brandes'e ait algoritmanın (Brandes 2001) dağıtık hale getirilmesi ile gerçekleştirilmiştir. Algoritmaları sayma ve birleştirme adını verdikleri iki ana bölümden oluşmaktadır. Birinci bölümde, dağıtık *BFS* ağaçları oluşturarak *APSP* işlemi yapılmakta ve kısa yolların sayısı hesaplanmaktadır. İkinci bölümde ise düğümlerin kısa yol kullanım oranları belirlenerek merkezilik değerleri hesaplanmaktadır. Kullandıkları model gereği mesaj boyutunun *O(log(n))* bit ile sınırlamışlardır. Bu mesaj boyu sınırından dolayı, merkezilik değerlerinde bir miktar hasiyet kaybı ile karşılaşmışlar ve bunu en aza indirebilmek için çalışmalar yapmışlardır.

NOBDC algoritmasının zaman karmaşıklığı *O(n)* round olarak verilmiştir, ama mesaj karmaşıklığı hakkında net bir bilgiye rastlanamamıştır. Bu çalışma Lehman ve arkadaşlarına ait çalışma (Lehmann and Kaufmann 2003) ile çok benzer alt yapıya sahiptir (zaman uyumlu olması ve *BFS* ağaçları ile *APSP* hesaplanması açısından). Bu nedenle mesaj karmaşıklığının $O(n^2m)$ olduğu kabuledilmiştir. Zaman karmaşıklığına göre oldukça ölçeklenebilir ve tüm çizge bazında değerlendirme yaptığı için güvenilir bir algoritmadır. Ama algoritmanın dinamiklik özelliği bulunmamaktadır.

Rastgele Yürüyüş Aradalılık Merkezilik (İng. Random Walk Betweenness Centrality) ($C_R(V)$): Rastgele yürüyüş aradalılık merkeziliği *(C_R),* sıvıların yayılım şekli benzeştirilerek işlem yapılan akış tabanlı bir ölçüttür. Bu yaklaşımda, kısa yol tabanlı merkeziliklerin aksine olası tüm yollar dikkate alınır. Bu nedenle C_R karmaşık ağ analizlerinde kullanılan en karmaşık ölçütlerden biridir.

Diğer aradalılık merkezilik ölçütlerinde olduğu gibi temel parametre yine bir düğüm üzerinden geçen mesaj sayısıdır. Bunu belirlemek için bir ya da birden fazla mesaj ağda belli bir süre dolandırılmaktadır. Mesajın düğümler arasında yayılımı ise genel yayın yerine sadece rastgele seçilen bir komşuya iletilmesi ile sağlanmaktadır. Belli bir zaman diliminde üzerinden en çok mesaj geçiren düğüm ise en merkezi düğüm, en yoğun düğüm, kesim ya da köprü noktası olarak kabul edilmektedir. İletişimde olası tüm yolların kullanılabileceği yaklaşımından dolayı, kısa yol tabanlı yaklaşımlardan daha farklı bir mesaj yayılımı gerçekleşmektedir. Bu özellik sayesinde aynı anda kritik/merkezi düğümlerin yedeklerinin de belirlenmesi mümkün olmaktadır. Geçmişte C_R benzeri çalışmalar olmakla birlikte, günümüz şekline Newman (Newman 2005; Borgatti and Everett 2006) tarafından getirilmiştir.

Newman, elektrik akım-akış aradalılığının sayısal olarak, rastgele yürüyüş aradalılığına eşit olduğunu ispat etmiştir. Sonuç olarak Newman'a ait merkezi C_R hesabı ağdaki tüm düğümler için matrisler kullanılarak hesaplandığında, algoritmanın en kötü durumdaki zaman karmaşıklığı $O((m+ n)n^2)$ dir (Newman 2005). Bu değer ölçeklenebilirliğin üst sınırlarındadır. Algoritma tüm çizge verisini kullanarak güvenilir sonuçlar üretmektedir ama dinamiklik özelliğine sahip değildir.

İkinci Sıra Merkezilik (İng. Second Order Centrality *SOC*) (Kermarrec et al. 2011): Kermarrec ve arkadaşları, *SOC* adını verdikleri yeni bir merkezilik yaklaşımı sundular. Bu yaklaşım, rastgele yürüyüş aradalılık merkezilik yaklaşımına dayanmaktadır ve dağıtık mimaride tasarlanmıştır. Ağdaki her düğüm kendi yerel verisini kullanarak bağımsız bir şekilde, kendi göreceli önem değerini (merkeziliğini) hesaplamaktadır. Bu çalışma olasılık tabanlı (İng. probabilistic) sonuçlar üreten bir yakınsama (İng. approximation) algoritmasıdır.

SOC çalışması, merkezi düğümlerin diğer düğümlere göre, rastgele yürüyüş mesajları tarafından daha çok ziyaret edileceği fikri üzerine kurulmuştur. Yazarlara göre, çalışmaları basit ve düşük maliyetli bir yaklaşımdır. Çünkü standart rastgele yürüyüş aradalılık merkezilikte olduğu gibi ağdaki tüm alıcı ve gönderici ikilileri dikkate alınmamaktadır. Basitçe her düğüm, kendisine gelen mesajların geliş zamanlarını kayıt etmektedir ve daha sonra bu sürelerin standart sapması hesaplanmaktadır. Standart sapma değeri ise ilgili düğümün *SOC* merkezilik değerini vermektedir.

Kermarrec ve arkadaşları, teorik ve benzetim çalışmaları ile yaklaşımlarının kritik/merkezi düğümleri belirlemede; dağıtık olarak çalışan ve tüm çizge verisini değerlendiren diğer çalışmalar kadar hassas olduğunu göstermişlerdir. Diğer önemli bir katkıları ise *SOC* yaklaşımının, çizgelerin imzasını (İng. graph signature) belirlemede kullanılabileceğini göstermeleridir.

SOC yaklaşımının merkezilik kısmı şu şekilde çalışmaktadır. Başlangıçta, herhangi bir düğüm tarafından üretilen bir rastgele yürüyüş mesajı ağa bırakılır. Ağda bir anda bir mesaj bulunabilir. Bu mesaj kaybolmadan, algoritma sonlanana kadar düğümler arasında iletilir. Mesaj iletimi kesintisiz, bekletilmeden, tarafsızlıkla ve rastgele seçilen bir komşu üzerinden yapılır. Mesaj alan her düğüm geliş sürelerini kayıt eder. Algoritma belirli bir süre sonra sona erdiğinde, her düğüm standart sapma (σ) hesabı yaparak, kendi merkezilik değerini öğrenmiş

olur. En düşük değere sahip düğüm en önemli düğüm olarak kabul edilmektedir. Mesajın iletileceği düğümün seçiminde, *1/n* eşitliği (İng. stationary distribution) kullanılarak olasılık hesabı yapılmaktadır. Çalışmada yeterince süre beklenirse, tüm düğümlerin mesaj alma olasılığının eşit olacağı sezgisel yaklaşımı kabul edilmiştir. Adil bir rastgele yürüyüş sağlayabilmek için *Metropolis–Hastings (Hastings 1970)* tekniğini kullanmışlardır.

SOC yaklaşımı, bir anda bir rastgele yürüyüş mesajı kullandığı için işleyiş zaman sıralıdır. Bu nedenle işlem süresi, diğer yaklaşımlara göre oldukça uzun olacak ve düğüm sayısına bağımlı olacaktır. Mesaj ve zaman karmaşıklığı da eşit olacaktır. Yazarlara göre, *SOC* 'ın yakınsama süresi/zaman karmaşıklığı $O(n^3)$ adımdır. Burada *n* çizgedeki düğüm sayısını temsil etmektedir. Yüksek karmaşıklık değerinden dolayı ölçeklenebilirliği üst sınırlarında bir çalışmadır. Algoritmanın dinamiklik özelliği bulunmamaktadır. Sonuçların hassasiyeti ise mesajların iletimde kaldığı süre ile doğrudan ilgilidir. Bu nedenle sonuçlar güvenilir değildir (Wehmuth et al. 2011a). Çalışmalarının sonunda Karemerrec ve arkadaşları, algoritmanın zaman maliyetini düşürmek amacıyla aynı anda birden fazla rastgele yürüyüş mesajı kullanımını önermişlerdir.

Dağıtık Akım Akış Aradalılık Merkezilik (İng. Distributed Current Flow Betweenness Centrality DUCKWEED) Algoritması (Lulli et al. 2015): Lulli ve arkadaşları *DUCKWEED* adında, dağıtık mimaride olan, zaman uyumsuz olarak çalışan ve büyük ağların analizi için tasarlanmış bir yakınsama algoritması sundular. Bu yaklaşım Newman ait (Newman 2005) rastgele yürüyüş aradalılık merkezilik yaklaşımına dayanmaktadır. Ağdaki her düğüm kendi yerel verisini kullanarak, komşuları ile mesajlaşarak ve D_ε adını verdikleri yakınsama değerine bağlı olarak bağımsız bir şekilde kendi göreceli önem değerini (merkeziliğini) hesaplamaktadır.

Algoritma, akım ve merkezilik hesaplama adında iki ana bölümden oluşmaktadır. İlk bölüm, kaynak düğümlerden hedef düğümlere gönderilen akımların yaratılması ile başlamaktadır. Akımların oluşturulmasında ve yayılmasında dedikodu (İng. gossip) protokollerinin bir bileşeninin kullanıldığı belirtilmiştir. Bu akımların dolaşımı anında, Kirchhoff kanunlarına göre her bir akım için her düğüm kendi potansiyel elektrik değerini hesaplamaktadır. Elektrik değerlerinde D_ε den daha az değişim olduğunda ilgili düğüm birinci aşamayı sonlandırmaktadır. İkinci aşamada ise her akıma ait bulunan potansiyel elektrik değerleri artırımlı (İng. incrementally) olarak toplanarak merkezilik değerleri

hesaplanmaktadır. Benzetim ortamında yaptıkları çalışmalar ile *DUCKWEED* algoritmasının, kullanılan akım sayısına ve seçilen D_ϵ değerine bağlı olarak kabul edilebilir yakınsama değerleri elde ettiklerini belirtmişlerdir.

Lulli ve arkadaşları, büyük ağlar için uygun ve ölçeklenebilir olduğunu belirttikleri çalışmaları için sadece benzetim ortamında kullandıkları çizgelere ait yakınsama değerlerini ve mesaj sayılarını gösteren grafikleri vermişlerdir. Algoritmalarının karmaşıklık değerlerine yönelik bir açıklama bulunamadığı için *DUCKWEED* algoritmasının ölçeklenebilir olup olmadığına karar verilememiştir. Algoritmanın dinamiklik özelliği yoktur ve güvenilirliği ise D_ϵ değeri ile yakından ilgilidir.

Köprü Bulucu (İng. Bridge Finder) Algoritması (Bradler et al. 2011): Bradler ve arkadaşları, aradalılık ya da rastgele yürüyüş aradalılık merkezilik yaklaşımlarına alternatif olabilecek dağıtık bir yakınsama algoritması sundular. Bu algoritma dedikodu yayılım tekniğini kullanmaktadır. Dedikodu tekniğinde, bir miktar bilginin ağda yayılması sağlanır bu amaçla başlatıcı düğüm tarafından üretilen rastgele bir sayısal değerin ağda yayılması sağlanmıştır.

Her düğüm, başlangıçta değeri sıfır olan bir yayılım değerine (d_{old}) sahiptir. Algoritmanın başında başlatıcı düğüm, d_{new} adı verilen bir değişkene, rastgele belirlenen bir kayan noktalı (İng. floating number) değer atar. Bu değeri komşu sayısının bir fazlasına bölerek yeni d_{new} değerini elde eder. Bu işlem aşağıdaki denklem ile gösterilmiştir.

$$\mathrm{d_{new}} = \frac{\mathrm{d_{old}} + \mathrm{d_{new}}}{|\Gamma(\mathrm{v})| + 1} \qquad (2.17)$$

Burada v, merkezilik değeri hesaplanacak olan düğümü temsil etmektedir.

Başlatıcı düğüm, d_{new} değerini komşularına gönderir ve bu değeri d_{old} değişkenine kopyalar. Yayılım mesajı alan bir düğüm, yukarıdaki denklem yardımıyla yeni yayılım değerini hesaplar ve komşularına iletir. Her mesaj gönderimi sayılarak ilgili düğümün tekrar (İng. iteration) sayısı hesaplanır. Yayılımın sonlanması için ϵ adı verilen ve kabul edilebilir değişmezlik (İng. tolerance) değerini ifade eden bir parametre kullanılır. Eğer, $|\mathrm{d_{old}} - d_{new}| \leq \epsilon$ ise ilgili düğüm yayılım yapmaz ve durumunu yakınsadı (İng. converge) olarak değiştirir. Yayılım işlemi, tüm düğümler yakınsadı durumuna gelene kadar devam etmektedir. Algoritmanın sonunda her düğüm, en hızlı k adet düğümün ismi olan

bir listeye sahiptir. Yazarlar benzetim çalışmalarında, ϵ = 0.02 ve k = 10 değerlerini kullanmışlardır. Sonuç olarak en küçük tekrar sayısı ile yakınsayan düğüm, ağdaki en hızlı yakınsayan düğümdür ve en kritik/merkezi düğüm olarak kabul edilmektedir. Ayrıca yazarlar, kritik düğümlere ait daha hassas değerler elde etmek için algoritmalarının birden fazla sayıda çalıştırılabileceğini belirtmişlerdir.

Bu yaklaşımdan başka, ortalama aradalılık (İng. Average betweenness) (C_{AB}), adını verdikleri yeni bir merkezilik yaklaşımı daha sunmuşlardır. Bu yaklaşımla amaçları kritik düğümlerin yedeklerini oluşturmaktır. İddialarına göre, bir düğüm eğer kritik bir düğüme komşu ise onun da aradalılık merkezilik değeri yüksektir. Bu nedenle merkezilik değeri hesaplanırken, o düğüme fazladan puan verilmelidir. Bu yaklaşım, aradalılık ile özvektör merkezilik yaklaşımlarının bir birleşimi gibi düşünülebilir. Hesaplanmasına yönelik denklem aşağıdaki gibidir.

$$C_{AB}(v) = \frac{C_B(v) + \sum_{W \in Neigh(v)} C_B(W)}{|\Gamma(v)| + 1} \qquad (2.18)$$

Burada v düğümü için $C_{AB}(v)$ ortalama aradalılık, $C_B(v)$ ise geleneksel aradalılık merkezilik değerini temsil etmektedir. $\Gamma(v)$ ise v düğümünün komşu kümesidir.

Yazarların iddiasına göre, algoritmaları var olan yakınsama algoritmalarına göre önemli ölçüde hızlı ve çok hassas değerler üretmektedir. Ayrıca mesaj karmaşıklığının da çok düşük olduğunu vurgulamışlardır. Ama elde ettikleri sonuçların, durağan ağlar üzerinde çalışan ve bütün çizge verisini kullanan geleneksel algoritmaların sonuçları ile kıyaslandığında daha az hassas oldukları görülmüştür. Bu nedenle sonuçların güvenilirliği düşüktür. Elde ettikleri sonuçların güvenilirliği algoritmada kullanılan tekrar sayısı ϵ ve k parametreleri ile yakından ilişkilidir.

Yazarlara göre, tüm düğümler aynı anda çalıştıkları için paralel bir işlem vardır ve algoritmalarının zaman karmaşıklığı $O(logn)$ dir. Burada n tüm ağdaki düğüm sayısıdır. Ama kesin zaman ve mesaj karmaşıklığı bilgilerini daha sonra açıklayacaklarını belirtmişlerdir. Bu çalışma karmaşıklıklarından dolayı ölçeklenebilir olarak kabul edilebilir. Fakat dinamiklik özelliğine sahip değildir. Yerel komşuluk verilerini kullanarak çalışan algoritma, yönlü/yönsüz ve karmaşık ağlar üzerinde işletilebilir niteliktedir.

Benlik Aradalılık Merkezilik (İng. Ego Betweenness Centrality *EBC*) Algoritması(Guidi et al. 2014): Guidi ve arkadaşları tarafından tasarlanan algoritma hem standart aradalılık hem rastgele yürüyüş aradalılık özelliklerine sahiptir. Bu çalışma bölüm 2.1.2.3 de detaylı olarak açıklandığı için burada tekrar edilmemiştir.

Sınıf Değerlendirmesi: Bir düğümün geçirgenlik (İng. permeability) oranı, o düğümün kullanılma sıklığına ya da kullanılma olasılığına bakılarak belirlenmektedir. Geçirgenlik algoritmaları diğer merkezilik algoritmalarına göre daha karmaşık ve daha çok kullanım alanına sahip algoritmalardır. Bu problemin çözümü için birçok olasılık (İng. stochastic) tabanlı ya da kararlı (İng. deterministic) algoritma sunulmuştur. Özelikle dağıtık mimarideki algoritmalar, karmaşıklık değerlerini düşürmek amacıyla, benlik ağı (İng. ego network) kullanımı gibi bazı hırslı (İng. greedy) yaklaşımlar kullanmıştır. Bu sayede yerel veri kullanımı ile genel ağa ait sonuçların elde edilmesi amaçlanmıştır. Bu durumda, elde edilen sonuçların güvenilirliği genellikle yerel ağ verisi ile genel ağ verisi arasındaki ilişkiye (İng. correlation) bağlı olmaktadır. Veriler arasındaki ilişki nedeniyle, kullanılacak merkezilik yaklaşımının hangi ağ yapısında ve hangi ağ büyüklüğünde güvenilir sonuçlar üretebildiğine dikkat edilmelidir. Bu sınıftaki algoritmaların çoğu bir atlama uzaklıktaki komşulardan oluşan (İng. *egocentric network*) yerel veriyi kullanmaktadır. Sadece *Restricted Stress Centrality* (P. Fekete et al. 2005) algoritmasında kullanılacak yerel veri miktarı kullanılan bir parametre yardımıyla değiştirilebilir yapıdadır.

Geçirgenlik sınıfındaki çalışmalardan, sadece Lehman ve arkadaşlarına ait olan genel amaçlı algoritma (Lehmann and Kaufmann 2003) ile *NODBC (Hua et al. 2016)* algoritması tüm çizge verisini değerlendirerek en iyiye yakın sonuçlar üretmiştir. Kararlı yapıdaki bu iki çalışma tüm çizge verisini değerlendirmelerinden dolayı diğerlerinden daha yüksek mesaj karmaşıklığına sahiptir. Anladığımız kadarı ile birbirlerine çok benzer olan bu iki çalışma arasındaki en büyük fark *NODBC* tarafından mesaj uzunluğuna getirilen sınırlamadır. Bu nedenle elde edilen sonuç, Lehman ve arkadaşlarınınkine göre daha düşük hassasiyete sahiptir. Diğer çalışmalar bekleme süresi, sonlanma değeri ve yerel veri miktarı gibi bir takım parametrelere bağlı olarak yaklaşık sonuçlar üretmektedirler. Dolaysı ile bu çalışmaların güvenilirliği kullanılan parametrelere yakından ilgilidir. *DUCKWEED* (Lulli et al. 2015) algoritmasının karmaşıklığı bilinmemekle birlikte *SOC* (Kermarrec et al. 2011) dışındaki yerel veri kullanan algoritmaların hepsi ölçeklenebilir düzeydedir.

Bildiğimiz kadarı ile *Restricted Stress Centrality (P. Fekete et al. 2005)* algoritmasının düğüm dağılımına yönelik getirdiği kısıtlamalardan dolayı, *SBet* (Oliveira et al. 2010) algoritmasının bir adet dolaşım ağacı kullanarak hesaplama yapması ve gecikme süreleri kullanmasından dolayı ve *SOC* algoritmasının ise sahip olduğu yüksek karmaşıklık düzeyinden dolayı; bu algoritmaların karmaşık ağların analizinde kullanımı uygun olmayabilir. Genellikle geçirgenlik sınıfı algoritmaları kullanarak köprü ya da kesim noktası gibi merkezi düğümlerin bulunması işleminde, bulunacak her bir merkezi düğüm için algoritmanın bir kez çalıştırılması gerekmektedir. *LBC (Nanda and Kotz 2008)* algoritmasının yazarları ise algoritmanın bir defa çalıştırılması ile çizgedeki gerekli tüm köprü/kesim noktalarının belirlenebileceğini iddia etmişlerdir. Bu yaklaşımın analiz sürecini hızlandıracağı açıktır. Fakat elde edilen sonuçların güvenilirliği kullanılan yerel veri ile genel çizge verisi arasındaki ilişkiye bağlıdır. Bu nedenle karmaşık ağlarda uygulanması zordur.

İncelenen akış tabanlı çalışmaların çoğu *(SOC, Bridge finder (Bradler et al. 2011)* ve dedikodu tabanlı *EBC* (Guidi et al. 2014)*)* olasılık (İng. stochastic) tabanlı yaklaşımlardır. Diğer çalışmalar ise kararlı (İng. deterministic) sonuçlar üreten yaklaşımlardır. Yedek merkezi düğümlerin kolaylıkla tespit edilebiliyor olması ise akış tabanlı merkezilik yaklaşımlarının en büyük avantajlarından biridir. Özellikle hata toleransı (İng. fault tolerant) gerektiren sistemlerin tasarımında kullanılabilmektedir. Yaptığımız çalışmalara göre, *EBC* (Guidi et al. 2014) algoritması dışında dinamiklik özelliğini destekleyen başka bir dağıtık aradalılık algoritması bulunamamıştır. İncelenen çalışmaların tarafsız bir şekilde değerlendirilebilmesi için *EBC* algoritmasının dinamiklik kısmına ait zaman ve mesaj karmaşıklıkları Tablo 2.3 e dâhil edilmemiştir.

Çizelge 2.3 Geçirgenlik sınıfı algoritmaların özet değerlendirme çizelgesi.

Sınıflar	Merkezilikler	Algoritmalar	Zaman Karmaşıklığı	Mesaj Karmaşıklığı	Ölçeklene-bilirlik	Dinamiklik	Güvenilirlik
Geçirgenlik	Gerginlik Merkezilik	Central Stress centrality algorithm	$O(n^3)$	_	Düşük	Hayır	Yüksek
		Decentralized algorithms for evaluating centrality in complex networks	$O(D_G)$	$O(n^2 m)$	Yüksek	Hayır	Yüksek
		Restricted stress centrality	$O(\delta)$	$O(nm)$	Yüksek	Hayır	δ değ. bağlı
	Aracılık Merkezilik	Central Betweenness centrality algorithm	$O(n^3)$	_	Düşük	Hayır	High
		EBC Algorithm with boradcast messages	$O(1)$	$O(n\|Ego\Gamma(v)\|)$	Yüksek	Evet	Düşük
		Sbet Algorithm	$O(D_G\, t_{delay})$	$O(m)$	t_{delay} değ. bağlı	Hayır	Düşük
		NODBC Algorithm	$O(n)$	$O(n^2 m)$	Yüksek	Hayır	Yüksek
		LBC Algorithm	$O(1)$	$O(\Delta nm_{Ego})$	Yüksek	Hayır	Düşük
		Decentralized algorithms for evaluating centrality in complex networks	$O(D_G)$	$O(n^2 m)$	Yüksek	Hayır	Yüksek
	Rastgele Yürüyüş Aracılık Merk.	Central Random walk betweenness centrality algorithm	$O((m+n)n^2)$	_	Düşük	Hayır	Yüksek
		DUCKWEED	Verilmemiş	Verilmemiş	Verilmemiş	Hayır	Düşük
		EBC Algorithm with gossip approach	$\Omega(\|Ego\Gamma(v)\|w)$	$\Omega(n\|Ego\Gamma(v)\|)$	Yüksek	Evet	Düşük
		Bridge finder algorithm	$O(logn)$	Verilmemiş	Yüksek	Hayır	Düşük
		SOC Algorithm	$O(n^3)$	$O(n^3)$	Düşük	Hayır	Düşük

2.1.3 Dağıtık merkezilik algoritmaların genel değerlendirmesi

Bu bölümde, yayınlanmış olan dağıtık merkezilik algoritmaları üzerine bir araştırma çalışması yapılmıştır. Algoritmaların, büyük ve karmaşık ağlarda kullanılabilirliği incelenmiştir. Bu açıdan bakılarak bulunabilen çalışmalar bazı kısıtlara göre değerlendirilmiş, kıyaslanmış ve sınıflandırılmıştır. Algoritmalar temel kullanım alanları dikkate alınarak bilinilirlik (İng. popularity), erişilebilirlik (İng. accessibility) ve geçirgenlik (İng. permeability) şeklinde sınıflandırılmıştır. Değerlendirme işleminde; algoritmaların kullanılabilirliği, ölçeklenebilirliği (İng. scalability), güvenilirliği (İng. reliability), dinamikliği (İng. dynamicity) mesaj ve zaman karmaşıklıkları gibi nitelikleri dikkate alınmıştır. Karmaşık ağlardaki düğüm sayısı *(n)* milyonlarca olabileceği için değerlendirmelerimizde n^3 ve üzeri

karmaşıklık değerleri ölçeklenebilirliğin üst sınırı olarak kabul edilmiştir. Karşılaştırmanın özeti Çizelge 2.1, 2.2 ve 2.3 de verilmiştir. Bildiğimiz kadarı ile sınıflar arası kıyaslama yapmak çok doğru olamayacaktır. Çünkü her sınıf, farklı bir amaç için geliştirilmiştir ve merkezilik ölçütleri göreceli olarak en önemli öğeyi belirlemektedir. Bu nedenle sınıflar arası yerine, sınıf içi değerlendirme yapmak daha anlamlı olacaktır. Değerlendirme çizelgelerinden de görülebileceği gibi dağıtık algoritmaların zaman karmaşıklıkları, merkezi algoritmalara göre oldukça düşüktür ve genellikle $O(n)$ ya da $O(D_G)$ düzeyindedir (akış tabanlı aradalılık yaklaşımları bunun dışında tutulmuştur). Bu nedenle ölçeklenebilir çalışmalardır. Buna karşılık mesaj karmaşıklıkları, dağıtık mimarinin doğası gereği ve uygulanan yaklaşıma bağlı olarak oldukça büyük değerlere ulaşabilmektedir. Yerel veri kullanarak bölgesel merkezilik belirleyen algoritmaların mesaj karmaşıklıkları diğerlerinden daha düşüktür (P. Fekete et al. 2005; Xing et al. 2010; ChongGun and Mary 2013; Montresor et al. 2013; Guidi et al. 2014; Ramos et al. 2014). Yerel veri kullanmasına rağmen tüm çizge verisini değerlendirerek işlem yapan bazı algoritmaların mesaj karmaşıklıkları bulunamamıştır. Bulunabilenlerin mesaj karmaşıklıkları ise $O(nm)$ veya üzeridir (Lehmann and Kaufmann 2003; Kermarrec et al. 2011; Hua et al. 2016).

Bilinilirlik sınıfına ait dağıtık algoritmalarda temel parametre düğüm derecesidir ve temel amaç dağıtıcı/toplayıcı (İng. hub) düğümleri bulmaktır (Ramos et al. 2014). Bunların sonuçları tüm çizge göz önüne alındığında, çoğunlukla daha yerel öneme sahip sonuçlardır. Bu nedenle sonuçların güvenilirliği düşüktür ve kullanılan yerel verinin çapı ile doğru orantılıdır. Ama düşük karmaşıklık değerlerinden dolayı bu algoritmaların ölçeklenebilirliği genellikle yüksektir. Bu algoritmalar ağ analizlerinde, sınıf merkezlerini (İng. core of clusters) ya da dağıtıcı/lider düğümleri bulmak gibi amaçlar için kullanılabilmektedir. Sadece, *k-core decomposition algorithm* (Montresor et al. 2013) isimli çalışmanın kullanım amacı diğerlerinden daha farklıdır. Burada amaç tek bir merkezi düğüm bulmak yerine, birbirine bağlı merkezi düğümlerin oluşturduğu bir kümeyi belirlemektir. Tüm çizge verisini değerlendiren bu çalışma bu sınıftaki en iyi dereceye sahip çalışmalardan biridir.

Erişilebilirlik sınıfına ait dağıtık algoritmalarda ise temel parametre yol ya da kısa yollardır. Temel amaç diğerlerine en yakın düğümü bulmaktır. Genellikle yerel veri kullandıkları için elde edilen sonuçlar bölgesel öneme sahiptir. Bu nedenle sonuçların güvenilirliği, kullanılan yerel verinin çapına ve kullanılan parametrelere bağlı olmakla birlikte genellikle düşüktür. Fakat sınırlı ya da yerel

veri kullanımı, işlem karmaşıklığı artan algoritmaların ölçeklenebilirliğini yükselten önemli etkenlerden biridir. Bu sınıfa ait algoritmaların çoğu yakınsama algoritmasıdır. Sadece Lehmann ve arkadaşlarına (Lehmann and Kaufmann 2003) ait olan genel amaçlı algoritma, kararlıdır ve ağ bazında güvenilir sonuçlar üretmektedir. Bu çalışmanın da zaman uyumlu olması ve yüksek mesaj maliyeti, karmaşık ağlara uygulanmasını zorlaştırmaktadır.

Geçirgenlik sınıfına ait dağıtık algoritmalarda da temel parametre, yol ya da kısa yollardır. Temel amaç ise üzerinden en çok trafik geçecek ya da geçme olasılığı olan düğümü bulmaktır. Bu düğümler çizgedeki köprü düğümler, kesim noktaları veya tıkanıklık noktaları vb. olabilir. Geçirgenlik algoritmaları, ayrıca düğümlerin sınıflandırılması amacıyla da kullanılabilmektedir (Newman and Girvan 2004). Bildiğimiz kadarı ile geçirgenlik sınıfına ait yaklaşımların artan bir kullanım oranı vardır. Bu sınıf içinde üç farklı grup bulunmaktadır. Bunlardan birincisi gerginlik merkezilik yaklaşımıdır ve sadece bir düğüm üzerinden geçen kısa yol sayısını dikkate alır. Bu alt gruba ait iki çalışma bulunabilmiştir. Bu gruptaki Lehman ve arkadaşlarına (Lehmann and Kaufmann 2003) ait olan genel amaçlı algoritma, kararlıdır ve güvenilir sonuçlar üretmektedir. Fakat bu algoritmanın zaman uyumlu olması ve yüksek mesaj maliyeti, karmaşık ağlara uygulanmasını zorlaştırmaktadır. İkinci grup aradalılık merkezilik yaklaşımıdır. Düğümlerin ağda kulanım oranları dikkate alınarak hesaplama yapılmaktadır. Bu gruba ait bulunan çalışmaların çoğu (Lehmann and Kaufmann 2003; Nanda and Kotz 2008; Oliveira et al. 2010; Guidi et al. 2014; Hua et al. 2016) yerel veri kullanarak işlem yapmışlardır. Genellikle bir atlama mesafedeki komşuluk verilerini kullandıkları için ölçeklenebilirlikleri yüksektir ama güvenilirlikleri düşüktür. Sadece Lehman ve arkadaşlarına ait (Lehmann and Kaufmann 2003) çalışmada yerel veri kullanılarak tüm çizge bazında sonuç üretilmiştir. Ölçeklenebilirliği ve güvenilirliği yüksek olan bir çalışmadır. Daha önce belirtildiği gibi, zaman uyumlu olması en önemli dezavantajıdır. Üçüncü grup ise rastgele yürüyüş aradalılık yaklaşımıdır. Daha önceki iki gruba ait algoritmalar sadece kısa yolları dikkate almaktadır. Bu grupta ise iletişimin olası her yol kullanılarak yapılabileceği kabul edilmektedir. Sıvı/elektrik akış modelini benzeştiren, oldukça etkin sonuçlar üretebilen, genellikle olasılık tabanlı ve karmaşık yaklaşımlardır. Bu gruptaki algoritmalar (Bradler et al. 2011; Kermarrec et al. 2011; Guidi et al. 2014; Lulli et al. 2015) karmaşık ağ analizleri için tasarlanmıştır. Ama kabul edilebilir sonuçlar üretmek için oldukça fazla mesaj kullanmak ve birden fazla sayıda çalıştırılmak zorunda kalınmışlardır. Bu nedenle ölçeklenebilirlikleri ve güvenilirlikleri bir takım parametrelere bağlıdır.

Araştırmalarımıza göre, *EBC* (Guidi et al. 2014) algoritmaları dışında dinamiklik özelliğine sahip dağıtık merkezilik algoritması bulunamamıştır.

2.1.4 Sonuç

Bildiğimiz kadarı ile polinomsal zamanda çözümü olmayan (İng. NP Complete) merkezilik algoritması bulunmamaktadır. Buna rağmen ağ analizlerinin karmaşıklığı ve kullanılan verinin büyüklüğünden dolayı algoritmaların zaman karmaşıklıkları(İng. complexities) oldukça önemlidir. Merkezi ve paralel olarak çalışan algoritmalar donanım sınırlarını zorlamaya başlamıştır. Bu nedenle dağıtık yaklaşımlar günümüzde giderek önem kazanmaktadır. Dağıtık sistemlerin önemli bir avantajı işlemci sayısıdır. Çünkü bu sayı ağdaki düğüm sayısına eşittir. Bu nedenle dağıtık algoritmaların zaman karmaşıklığı genellikle çok düşüktür ve ölçeklenebilirlikleri yüksektir. Dağıtık sistemlerin en önemli dezavantajı ise aratan mesaj karmaşıklığıdır. Bu yüzden pek çok merkezilik algoritmasında, mesaj maliyetlerini düşürmek ve hız artışı sağlamak için açgözlü (İng. greedy) yaklaşımlar kullanılmıştır. Bu yaklaşımlar sınırlı veri kullanmak, veri örneklemesi yapmak veya olasılık tabanlı yöntemleri kullanmak olarak listelenebilir. Sınırlı veri genellikle benlik ağı (İng. ego network) olarak adlandırılan, ilgili düğümün yakın çevresindeki kenar ve düğümlerden oluşan, genel ağın küçük bir kısmıdır. Sınırlı veri kullanmanın en büyük dezavantajı ise algoritmaların güvenilirliğini azaltabilmesidir. Eğer yerel veri ile genel çizge verisi arasında bir ilişki (İng. correlation) varsa ve hassas sonuçlar üretmek yerine sadece önemli düğümlerin sırası yeterli ise o zamana sınırlı veri kullanımı ile kabul edilebilir sonuçlar üretilebilmektedir. Aksi takdirde hassas sonuçlar üretebilmek için tüm çizge verisini değerlendirerek sonuç üreten yaklaşımlar kullanılmalıdır (Lehmann and Kaufmann 2003; Montresor et al. 2013). Bu durumda veri büyüklüğünden dolayı oldukça etkin algoritmalara ihtiyaç duyulmaktadır. Ayrıca araştırmalarımıza göre, *EBC* algoritmaları dışında dinamiklik özelliğine sahip dağıtık merkezilik algoritması bulunamamıştır. Bunun nedeni, merkezilik yaklaşımlarının çizge analizlerinde birer araç olarak görülmesi ve bir bütünün alt parçası olarak değerlendirilmesinden dolayı olabilir.

Var olan çalışmalar incelendiğinde, sadece 19 adet dağıtık merkezilik tabanlı algoritma bulunabilmiştir. Bu sayı, bu alanda yapılan çalışmanın ne kadar az olduğunun bir göstergesidir. Bu nedenle bu alanda, bir tez çalışması yapmaya çalışmaya karar verilmiştir ve ilk olarak sorgu işleme sürecinin kaynak atama aşamasında kullanılabilecek merkezilik tabanlı algoritmalar geliştirilmiştir.

3. ÖNERİLEN ALGORİTMALAR

Bu çalışma dağıtık sorgu işleme sistemlerindeki kaynak atama probleminin çözümü için merkezilik yaklaşımını kullanarak, farklı ve etkin çözümler üretmeyi amaçlamaktadır. Merkezilik algoritmaları genellikle bir çizge/ağ içinde göreceli olarak önemli olanların belirlenmesi için kullanılmaktadır. Bu güne kadar tanımlanmış birçok merkezilik türü vardır. Bir ya da birden fazlası aynı anda bir ağ üzerindeki analiz çalışmasında kullanılabilmektedir. Bununla birlikte belirli bir ağ analizi için seçilecek merkezilik türü, tamamen uygulamanın ihtiyaçları doğrultusunda belirlenmektedir. Kaynak atama uygulaması açısından bakıldığında da seçilebilecek birçok merkezilik yaklaşımı bulunabilir. Uygun merkezilik yaklaşımının seçiminde dağıtık merkezilik algoritmaları üzerine yapılan araştırma çalışması temel alınmıştır. Bu araştırma çalışmasında, var olan merkezilik yöntemleri incelenerek ve merkezilik ölçüm yöntemlerinin temel amaçları dikkate alınarak bir sınıflandırma yapılmıştır. Bunlar; *i.* Bilinilirlik (İng. popularity), *ii.* Erişilebilirlik (İng. accessibility) ve *iii.* Geçirgenlik (İng. permeability) tabanlı merkeziliklerdir. Araştırma çalışmasına dayanarak ve farklı merkezilik türlerinin etkilerini inceleyebilmek için yukarıda bahsedilen her sınıfa ait bir merkezilik algoritması önerilmiştir. Seçilen algoritmaların ait oldukları sınıfın iyi birer örneği olmasına özen gösterilmiş ve kaynak atama ihtiyaçları doğrultusunda gerekli düzenlemeler yapılmıştır.

Bu tez çalışmasının en temel düzenleme kuralı ve aynı zamanda bu alandaki çalışmalara olan katkısı "Sınırlı Merkezilik Yaklaşımı" dır (İng. Limitted Centrality Approach). Bu yaklaşım Lemma-1.1 de açıklanmış ve ispat edilmiştir. İkinci düzenleme kuralı ise "Uzaklık ve Erişim Zamanı"dır. Bu yaklaşımda Lemma-1.2 de açıklanmış ve ispat edilmiştir.

Lemma-1.1 Sınırlı Merkezilik: Sorgu işleme sisteminin doğası gereği çizgedeki tüm düğümlere ait merkezilik işlemi yapmaya gerek yoktur. Sadece ilgili düğümlere ait merkezilik hesaplarını yapmak sorgu işleme algoritmaların zaman, mesaj karmaşıklıklarını (İng. complexities) azaltacak ve doğru adayların belirlenmesini sağlayacaktır.

İspat: Uzaklık ya da en kısa yol yaklaşımını kullanan yakınlık ve aradalılık merkezilik algoritmaları, normal şartlarda tüm düğümlerden tüm düğümlere en kısa yol (İng. All Pair Shortest Path *APSP*) hesaplamalarını yapmalıdır. *APSP* yapan algoritmaların çoğu bir bilgisayar üzerinde çalışan merkezi (İng.

centralized) çözümlerdir ve zaman karmaşıklıkları $O(n^3)$ dür (Brandes 2001; Lehmann and Kaufmann 2003; Newman 2005; Guidi et al. 2014). Sorgu işleme sistemlerinde, veri akışı sadece veri kaynaklarından sorgu işleme birimine doğru olmaktadır. Veri kaynakları sadece *Vc* düğümler olabilirken, sorgu işleme birimi alt çizge içindeki herhangi bir düğüm (*Vc* ya da *Vuc*) olabilir. Bir başka deyişle, her zaman en kısa yolun bir ucu mutlaka *Vc* düğümüyken, diğer ucu *Vc* ya da *Vuc* düğüm olacaktır. Bu nedenle merkezilik değerlerinin hesaplanmasında tüm düğümler arasındaki kısa yolların bulunması yerine, *Vc* düğümlerinden *Vc* düğümlerine ve *Vc* düğümlerinden *Vuc* düğümlerine olan tüm kısa yolların bulunması yeterli olacaktır. *Vuc* düğümlerinden *Vuc* düğümlerine olan kısa yollar kullanılmayacağı için ihmal edilmelidir. Bu yolla sadece sonuca etki eden bileşenlerin merkezilik değerlerini hesaplayarak doğru adayların seçimi sağlanacaktır. Bu yaklaşıma sınırlı merkezilik adının verilmesinin nedeni budur. Böylece alt çizgedeki *Vuc* düğüm sayısına bağlı olarak merkezilik hesaplama algoritmalarının karmaşıklılarında azalma olacaktır.

Aşağıdaki denklemden de görülebileceği gibi, tüm düğümler arası hesaplanacak kısa yol sayısı, her zaman sınırlı merkezilik algoritmasının hesaplayacağı kısa yol sayısından fazla olacaktır.

$$P(|Vc|+|Vuc|,2)_{APSP} \geq [P(|Vc|+|Vuc|,2) - P(|Vuc|),2)]_{lim.cent} \qquad (3.1)$$

Lemma-1.2 Uzaklık ve Erişim zamanı: Kenar ağırlıklı çizgelerde, ağırlıklar düğümler arası iletişim süresi ya da uzaklık olarak düşünülebilir ve bu sayede kısa yol hesaplamaları yapılabilir.

İspat: Çalışılan çizgelerin kenar ağırlıklı olduğu önceden varsayılmıştır. Bu ağırlık değerlerini düğümler arasındaki veri iletim süresi olarak ya da tersini ilgili kenarın kanal genişliği (İng. bandwidth) olarak kullanabiliriz. Bu durumda bir kenarın ağırlık değeri ne kadar büyükse veri iletim süresi o kadar uzun, ne kadar küçükse veri iletim süresi o kadar kısa olacaktır. Bu dönüşüm aşağıda denklem ile sağlanmaktadır.

$$\text{Kanal Genişliği}(u,v) = \text{En Büyük Kanal Genişliği} / \text{Kenar Ağırlığı}(u,v) \qquad (3.2)$$

Bu özellik kısa yolların hesaplanmasında kullanılabilir. Veri taşırma (İng. flooding) yönteminde; kök düğüm veri paketlerini taşırdığında bir düğümün, bir kök düğüme ait aldığı ilk paket en hızlı ulaşan pakettir. Dolayısı ile en kısa yolu

kullanan paket olacaktır. Bu sayede dağıtık programlama mimarisinde en kısa yol hesapları yapılabilmektedir.

Önerilen algoritmalar:

i. Yakınlık Merkezilik Tabanlı Kaynak Atama (İng. Closeness Centrality Based Candidate Generation Algorithm CCBC) algoritması. Alt çizgedeki tüm düğümlerin sınırlı yakınlık merkezilik değerlerini hesaplar.

ii. *k*-uzaklık Merkezilik Tabanlı Kaynak Atama (İng. *k*-distance Centrality Based Candidate Generation Algorithm KDCBC) algoritması. İndirgenmiş alt çizgedeki tüm düğümlerin *k*-uzaklık *Vc* düğüm komşuluk değerlerini hesaplar. Burada *k* değeri sıfır ile alt çizge çapı arasında bir değer alabilir.

iii. Aradalılık Tabanlı Kaynak Atama (İng. Betweenness Centrality Based Candidate Generation Algorithm BCBC) algoritması. İndirgenmiş alt çizgedeki tüm düğümlerin sınırlı aradalılık merkezilik değerlerini hesaplar.

Önerilen algoritmaların genel özellikleri:

i. Dağıtık (İng. distributed),

ii. Bir kaynaktan başlatılmakta (İng. single source initiator), başlatıcı düğüm bir *Vc* düğüm ve aynı zamanda sorgu sahibi (İng. query owner),

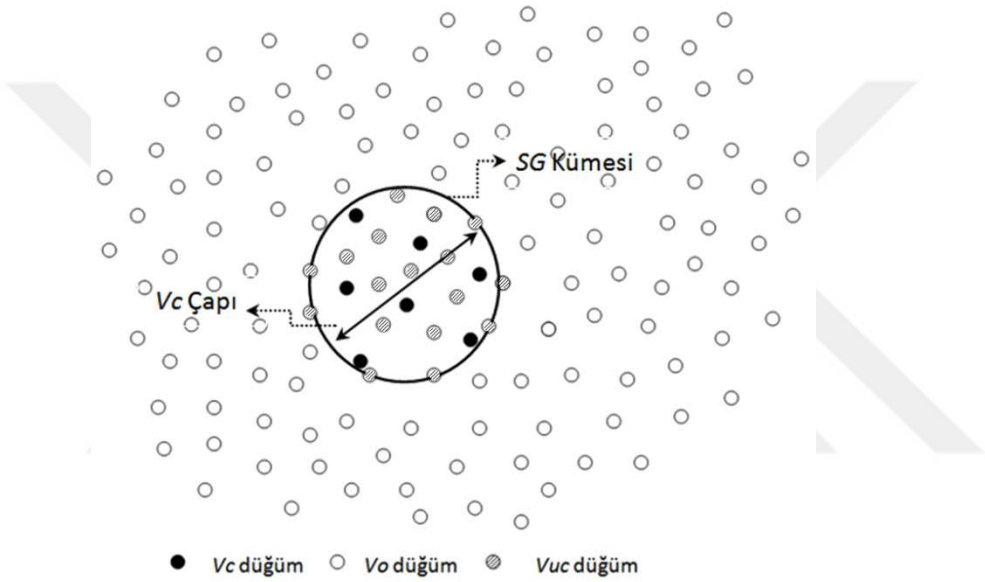
iii. Zaman uyumsuz (İng. asynchronous).

3.1 Alt Çizgenin Belirlenmesi

Dağıtık sorgu işleme sisteminde ana çizgedeki düğüm sayısı veri kaynağı olan düğüm sayısına oranla çok büyüktür. Bu nedenle sadece ilgili düğümler ve onların arasındaki sıradan düğümlerden oluşan bir alt çizge (*SG*) belirlemek faydalı olacaktır. Belirlenecek *SG;* ana çizgeyi *G* ile ifade edecek olursak, $G=(V,E)$ *ve* $SG=(V',E')$ iken, $V' \subseteq V$ dir ve $E' \subseteq E$ dir. $SG \subseteq G$ şartını sağlamalıdır ve elde edilecek *SG* bağlantılı bir çizge olmalıdır.

Önerilen kaynak atama algoritmalarının karmaşıklık düzeyleri bu yaklaşım sayesinde indirgenebilir. Alt çizgedeki düğümler veri kaynağı olan *Vc* ve veri kaynağı olmayan *Vuc* düğümler olarak tanımlanan düğümlerden oluşacaktır. *SG* belirlemenin temelinde yatan sezgisel yaklaşım, bir birine yakın düğümler arasındaki iletişim maliyetinin düşük olacağı şeklindedir. Dolayısı ile seçilecek en uygun adaylar, *Vc* düğümlerin hepsini içeren bir dairenin içindeki düğümlerden

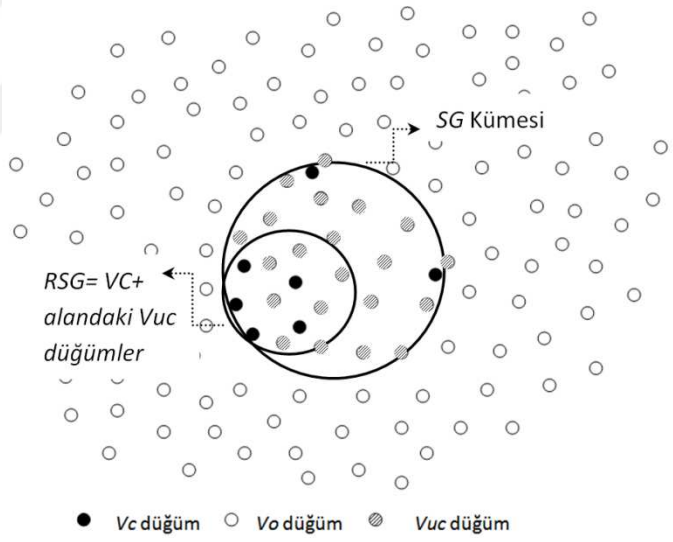
bazıları olacaktır. Bir başka deyişle, dairenin dışındaki kalan düğümlerin dairenin içinde kalan düğümlerden daha kötü başarı elde edeceği şeklindedir. *Vc* düğüm sayısı bilinmekle birlikte, *Vuc* düğümlerinin sayısı tamamen ana çizgenin yoğunluğuna ve *Vc* düğümlerin dağılımına (İng. topology) bağlıdır. *Vc* düğümler arasındaki en büyük uzaklık, kaynak keşfi aşamasında hesaplanabildiği için biliniyor kabul edilmiştir. Bu uzaklık, kullanılacak alt çizgenin çapıdır ve bu çap içinde kalan düğümler de istenen *SG* kümesini oluşturmaktadır. Aşağıdaki şekillerde düğümlerin yerleşimi ve düğümleri birbirlerine bağlayan kenar ağırlıkları dikkate alınarak yapılmıştır. Bir başka deyişle düğümlerin birbirlerine göre olan fiziksel konumları ile bir ilgisi yoktur.



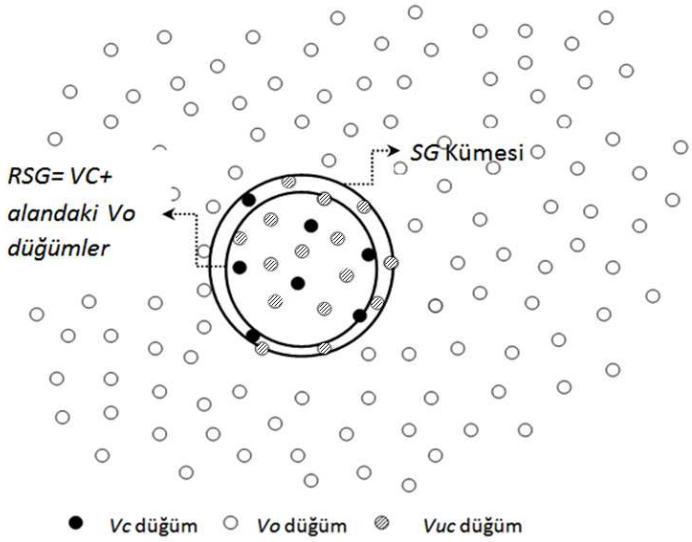
Şekil 3.1 Düğümlerin birbirlerine olan uzaklıklarına göre olan durumlarını göstermektedir.

SG kümesini oluşturmak için CCBC adıyla önerdiğimiz yakınlık merkezilik algoritmasından faydalanılmıştır. Çünkü bu algoritma, bizim amacımıza uygun *SG* kümesi tespitinde kullanılabilecek yaklaşımlar içinde, bildiğimiz kadarı ile en hızlı olan dağıtık algoritmadır. Bu algoritmanın çalıştırılması sonucu, *Vc* düğümlerin çapı içinde bulunan düğümlerin tamamı tespit edilebilmektedir. Daha önce genel özelliklerini tanımladığımız çizgeler kullanılarak yapılan benzetim çalışmalarının sonucu göstermiştir ki elde edilen *SG* boyutu, çizge yoğunluğu ve *Vc* düğüm çapı ile yakından ilişkilidir. Çizge çapı büyüdükçe (*SG* boyutu/Çizge boyutu) oranı küçülmesine rağmen, karmaşık ve internet gibi milyonlarca düğüm içeren büyük ağlar düşünüldüğünde, elde edilen *SG* boyutu oldukça büyük olabilmektedir. Dolayısı ile *SG* boyutunu daha aşağı çekmek amacıyla indirgenmiş alt çizge (*RSG*) adında bir yaklaşım daha önerilmiştir. *RSG* kümesi *Vc* düğümler ile *SG* içinde kalan bir kısım *Vuc* düğümden oluşmaktadır ve

RSG⊆*SG*⊆*G* şartını sağlamalıdır. Bu yaklaşım da *SG* belirlemede kullanılan yakınlık kavramına dayanmaktadır ve *SG* içinde *Vc* düğümlerin çoğunun öbeklendiği bir bölge varsa en uygun adaylar o bölge içinden seçilmelidir. *RSG* kümesinin *Vc* ler dışındaki elemanları, yakınlık merkezilik değerleri tüm *Vc* düğümlerin birbirlerine olan yakınlık merkezilik değerlerinin ortalamasına eşit ya da küçük değere sahip olan *Vuc* düğümlerden oluşmaktadır (Bkz. Denklem 1.3). Açıklamalardan da anlaşılacağı gibi *RSG* kümesinin belirlenmesinde *SG* nin belirleme aşamasında elde edilen yakınlık veriler kullanılmaktadır. Bu nedenle sisteme getirdiği maliyet ihmal edilebilir düzeydedir. *Vc* düğümlerin öbeklenmesine bağlı olarak oluşabilecek *RSG* kümesinin boyutu değişecektir (Bkz. Şekil 3.2 ve 3.3). Farklı çizgeler üzerinde yapılan benzetim çalışmaları göstermiştir ki, bu yaklaşımla elde edilen *RSG* kümesinin boyutu *SG* kümesin boyutundan çok daha küçüktür. *RSG* ve *SG* kümeleri kullanılarak belirlenen aday listeleri kıyaslandığında ise listelerin ilk bölümleri arasında eşitlikler ya da çok az farklılıklar olduğu gözlenmiştir.



Şekil 3.2 *Vc* düğümler bir bölgede öbeklendiğinde oluşan *RSG* kümesi.



Şekil 3.3 *Vc* düğümler bir bölgede öbeklenmediğinde oluşan *RSG* kümesi.

RSG kümesi, diğer algoritmalara göre çok daha büyük zaman ve mesaj karmaşıklığı olan KDCBRA ve BCBRA isimli algoritmalar tarafından kullanılmaktadır. Bu sayede daha verimli çalışmaları sağlanmıştır.

Bu açıklamalar doğrultusunda, alt çizge belirleme işleminde kullanılan CCBC algoritması, Algoritma 3.1 de detaylı olarak anlatılmıştır.

3.2 Yakınlık Merkezilik Tabanlı Kaynak Atama (İng. Closeness Centrality Based Candidate Generation Algorithm CCBC) Algoritması

Bu algoritma, erişilebilirlik sınıfına ait bir merkezilik algoritmasıdır. Bu yaklaşımda esas amaç, tüm veri kaynaklarına (*Vc*) en yakın düğümü bulmaktır. Bu sayede, veri kaynaklarından gönderilen veri paketleri en kısa yolu kat edecek ve sorgulama işlemine ait ağ trafik en aza indirgenebilecektir. Algoritma sınırlı yakınlık merkezilik hesabı yaparak atanacak kaynaklar için bir aday listesi oluşturmaktadır. Sınırlı yakınlık merkezilik hesaplaması yapılmasının nedeni Lemma-1.1 de açıklanmıştır. Bu yaklaşımla *Vc* düğümlerinden *Vc* düğümlerine ve *Vc* düğümlerinden *Vuc* düğümlerine olan kısa yollar kullanılarak, $|Vc|+|Vuc|$ düğüm sayısı kadar yakınlık merkezilik değerleri hesaplanmaktadır. Algoritmanın karmaşıklıklarını (zaman ve mesaj) düşürmek amacıyla, ana çizgenin bir bölümünü oluşturan ve alt çizge diye isimlendirdiğimiz alanda çalışmaktadır. Ayrıca bu algoritma, alt çizge ve indirgenmiş alt çizge kümelerine ait düğümlerin belirlenmesi için de kullanılmaktadır.

Bu doğrultuda hazırlanan algoritma aşağıdaki gibidir.

Mesaj tipleri:

i. **START_VC_CLOSNESS:** Başlatıcı *(Vi)* düğümden, *Vc* düğümlere alt çizge belirleme ve aynı zamanda alt çizgedeki yakınlık merkezilik değerlerini hesaplama işlemini başlatmaları için doğrudan gönderilen mesajdır.

ii. **VC_PROBE:** Taşırma (İng. flooding) yoluyla en kısa yolların belirlenmesi amacıyla, *Vc* ya da *Vo* düğümleri tarafından, *Vc* ya da *Vo* düğümlerine gönderilen mesajdır. Çizgede *Vc* çapı kadar yayılacaktır.

iii. **VX_CLOSNESS:** Başlatıcı düğümü bilgilendirmek amacıyla *Vc* düğümler tarafından doğrudan gönderilen mesajdır. İlgili düğüme ait yakınlık değeri içerir.

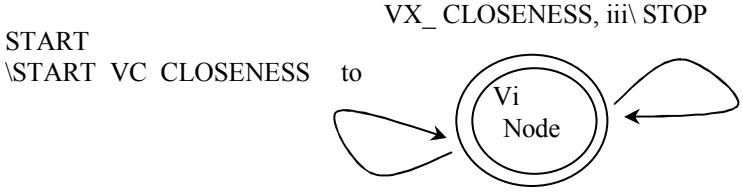
Çizelge 3.1 Alt çizge belirleme algoritmasının zaman ve mesaj karmaşıklıkları.

Mesaj Adı	**Zaman Karmaşıklığı**	**Mesaj Karmaşıklığı**
Start_Vc_Closeness	O(d)	$\Theta(Vc)$
Vc_Probe (Flooding)	O(d)	$O(4*Vc*E_s)$
Vx_Closeness	O(d)	$\Theta(N_s)$
CCBC **Algoritması**	O(d)	$O(Vc + 4VcE_s + N_s)$
Sonuç	**O(d)**	$\mathbf{O(E_s + N_s) \leq O(N + E)}$

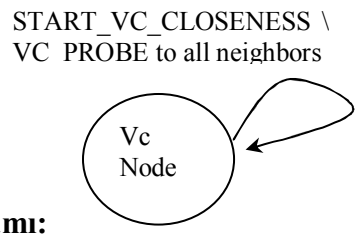
Burada *d* alt çizge çapını, E_s alt çizgedeki kenar sayısını, N_s alt çizgedeki düğüm sayısını temsil etmektedir.

Yakınlık merkezilik tabanlı kaynak atama algoritması durum diyagramı:

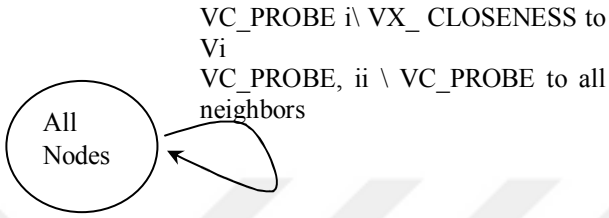
***Vi* düğümü durum diyagramı:**



***Vc* düğümü durum diyagramı:**



Tüm düğümlerin durum diyagramı:



Şartlar:

i. Eğer ilgili *Vc* ye ait ilk defa VC_PROBE mesajı alındı ve tüm *Vc* düğümlerden de VC_PROBE mesajı alındı ise kendi yakınlık merkezilik değeri VX_CLOSENESS mesajıyla *Vi* düğüme gönderilir.

ii. Eğer ilgili *Vc* ye ait ilk defa VC_PROBE mesajı alındı ve maksimum mesaj gönderim uzaklığı aşılmadı ise alınan mesaj, mesajın göndericisi ve *Vi* düğüm dışındaki tüm komşulara taşırma yolu ile iletilir. Eğer ilgili *Vc* ye ait ilk defa VC_PROBE daha önce alındı ve maksimum mesaj gönderim uzaklığı daha öncekinden küçük ise merkezilik değeri güncellenir ve alınan mesaj, mesajın göndericisi ve *Vi* düğüm dışındaki tüm komşulara taşırma yolu ile iletilir.

iii. Eğer diğer tüm *Vc* düğümlerden de VX_ CLOSENESS mesajı alındı ise aday listesi oluşmuş demektir, algoritma sonlandırılır.

Algoritma 3.1 Yakınlık Merkezilik Tabanlı Kaynak Atama Algoritması (CCBCG)

Amaç: *Vc* çapı kadar mesafede içinde, tüm *Vc* düğümlerine ulaşabilen düğümleri bularak *SG* kümesini oluşturmak, bunlara ait sınırlı yakınlık merkezilik değerini hesaplamak ve *RSG* kümesini belirlemek.

Girdiler: *Vc* düğüm kümesi, alt çizge çapı, başlatıcı düğüm numarası (İng. node id).

Çıktılar: Alt çizge düğüm kümesi (*SG*) (*Vc* ve *Vuc* düğümleri), alt çizgedeki düğümlere ait sınırlı yakınlık merkezilik değerleri, indirgenmiş Alt çizge düğüm listesi (*RSG*).

Sonlanma (İng. termination condition**):** Başlatıcı düğüm, tüm *Vc* düğümlerden yakınlık merkezilik değerini aldığında algoritma sonlanır. Bu süre zarfında *SG* içinde kalan diğer düğümlerin (*Vuc*) de merkezilik değerlerini ilettiği kabul edilir.

Vi **node:**

1: *Vc_Counter* ← *0*
2: **send** START_VC_CLOSNESS message to all *Vc* nodes
3: *Vi* node starts to **wait** VX_CLOSENESS message of nodes
4: **Upon received** a VX_ CLOSENESS message from a node:
5: **add** sender node limited closeness centrality value to Closeness_List
6: **if** sender node is a Vc node **then increment** *Vc_Counter*
7: **if** *Vc_Counter* is equal to number of *Vc* node **then**
8: **calculate** average of *Vc* node closeness
9: **sort** and **list** all Closeness_List as *SG* set
10: **list** some nodes of Closeness_List which are small or equal to average of *Vc* as RSG set
11: **end if**
12: **end of CCBC**

Other **nodes:**

1: **Upon received** a START_VC_CLOSNESS message from *Vi* node:
2: **if** distance of message is not greater than Subgraph diameter **then**
3: **send** VC_PROBE message to all neighbors
4: **Upon received** VC_PROBE message:
5: **if** it is first VC_PROBE message of related Vc node **then**
6: **increment** *Vc_Probe_Counter*
7: **Add** distance of message to my closeness value
8: **if** distance of message is not greater than Subgraph diameter **then**
9: **send** VC_PROBE message to all neighbors except root and sender
10: **if** *Vc_Probe_Counter* is equal to number of Vc node **then**
11: **send** a VX_CLOSENESS message to *Vi* node with closeness value.
12: **else**
13: **if** distance of msg. is smaller than registered message of related *Vc* node **then**
14: **update** my centrality value
15: **if** distance of message is not greater than Subgraph diameter **then resend** VC_PROBE message to all neighbors except root and sender
16: **end if**
17: **end if**

3.3 *k*-Uzaklık Merkezilik Tabanlı Kaynak Atama Algoritması (İng. *k*-Distance Centrality Based Candidate Generation Algorithm KDCBC)

Bu algoritma, bilinirlik sınıfına ait bir merkezilik yaklaşımıdır. Bu yaklaşımda düğümlerin bilinirlik derecesi ya da önemi, düğümlerin sahip oldukları komşu sayısına bakılarak belirlenmektedir. Bu çalışmadaki temel amaç, alt çizgede her bir düğümün kendisinden *k* kadar uzaklıkta kaç tane *Vc* komşusu olduğunu bulmaktır. En çok *Vc* komşusu olan düğümlerin aday olarak seçilecektir. Böylece veri kaynaklarından gönderilen veri paketleri daha az yol kat edecek ve sorgulama işlemine ait ağ trafiği en aza indirgenebilecektir. Bu algoritmada *k* değeri başlangıçta bilinmesi gereken kritik öneme sahip bir parametredir. Çünkü bu değer *Vc* düğümlerin alt çizge içindeki dağılımı ile doğrudan ilgilidir ve her farklı *Vc* kümesi için de farklı olacaktır. Ancak *k* değeri alt çizge yarıçapına eşit olduğunda bu yaklaşıma göre en uygun adaylar seçilebilecektir. Bunun nedeni Lemma-3.1 de açıklanmış ve ispat edilmiştir. Bu algoritma çalıştırılmadan önce alt çizge belirleme algoritması (CCBC) çalıştırılmalı ve indirgenmiş alt çizge kümesi belirlenmiş olmalıdır. Çünkü KDCBC algoritmasının mesaj karmaşıklığını düşürmek amacıyla *RSG* kümesi tanımlanmıştır. Algoritmanın çalışması sonucu, *RSG* kümesindeki her düğümün *k*-uzaklık komşuluk değerleri sıralanarak aday listesi oluşturulmaktadır.

Lemma-3.1 *k* parametresinin değeri: *k*-uzaklık merkezilik yaklaşımını bu çalışmanın kullanım amacı doğrultusunda değerlendirdiğimizde, *k* nın alabileceği en küçük ve en uygun değer *Vc* çapının (alt çizge çapının) yarısı olmalıdır. Bu sayede alt çizgenin merkezinde bulunan ya da merkeze en yakın düğümler aday olarak belirlenebilir.

İspatı: Bu parametre ile amaçlanan *k* kadar mesafe içinde, en çok *Vc* düğüme ulaşabilen ya da mümkünse tüm *Vc* düğümlere ulaşabilen düğümü bulmaktır. Bir başka deyişle en iyi adaylar en çok *Vc* düğüme, en küçük *k* değeri ile ulaşabilen düğümlerdir. Bu nedenle *k* nın en küçük değeri, alt çizgenin en merkezinde bulunan düğümün dışmerkezlik (İng. eccentricity) değerine eşittir. Bu değer aynı zamanda alt çizge çapının (*Vc* çapının) yarısına eşittir. Alt çizge merkezinde olamayan herhangi bir düğümün tüm *Vc* düğümlere ulaşabilmesi için kullanması gereken *k* değeri, alt çizge çapın yarısı ile alt çizge çapı arasında bir

değer olmalıdır. Yani merkezde olan düğüme göre daha büyük bir *k* değeri kullanmalıdır.

Aşağıda *k* nın alacağı değerlere göre *k*-uzaklık merkezilik değerleri daha net gösterilmiştir.

k parametresi en az sıfır, en fazla alt çizge çapı (biliniyor kabul edilen ve *Vc* düğümler arasındaki çap) (Vc_d) kadar bir değere sahip olmalıdır ($0 \leq k \leq Vc_d$). Bunun dışındaki *k* değerleri anlamsızdır.

k=0 iken : Eğer düğümün kendisi *Vc* ise *k*-uzaklık merkezilik değeri sadece bir olacaktır. Çünkü sadece kendine ulaşabilecektir.

Eğer düğümün kendisi *Vuc* ise *k*-uzaklık merkezilik değeri sıfır olacaktır.

$k<Vc_d/2$ iken: Düğümlerin *k*-uzaklık merkezilik değerleri farklılık gösterecek ama hiç bir düğüm tüm *Vc* lere ulaşamayacağı için merkezilik değerleri anlamasız olacaktır.

$k \geq Vc_d$ iken: Tüm düğümlerin *k*-uzaklık merkezilik değeri çizgedeki *Vc* düğüm sayısına eşit olacaktır. Adaylar arasında sıralama yapmak mümkün olmayacaktır.

$Vc_d/2<k<Vc_d$ iken: *k* değeri Vc_d değerine yaklaştıkça tüm *Vc* düğümlere ulaşabilen düğüm sayısı artacağı için *k* değeri büyüdükçe seçicilik azalacaktır.

$k=Vc_d/2$ iken: Sadece en merkezdeki düğüm tüm *Vc* lere ulaşabilecektir. Alt çizge merkezinde bulunan ya da merkeze yakın düğümlerin *k*-uzaklık merkezilik değerleri diğerlerinden daha büyük olacaktır.

Sonuç olarak bu çalışma bir eniyileme çalışmasıdır. Bu nedenle *k* nın en küçük ve en seçici olan değeri kullanılmalıdır, yani $k=Vc_d/2$ olmalıdır. Bu sayede en çok *Vc* düğüme ulaşabilen, alt çizgenin merkezindeki düğümler bulunabilecektir.

Mesaj Tipleri:

i. START_KDIST: *k*-uzaklık komşuluk hesaplama işlemlerini başlatmak amacıyla, başlatıcı düğümden *Vuc* ve *Vc* düğümlerine gönderilir.

ii. KDIST_JOIN: *k*-uzaklıktaki komşuları bulmak için taşırma yöntemiyle, *Vuc* veya *Vc* düğümlerinden *Vc, Vuc* veya *Vo* düğümlerine gönderilir.

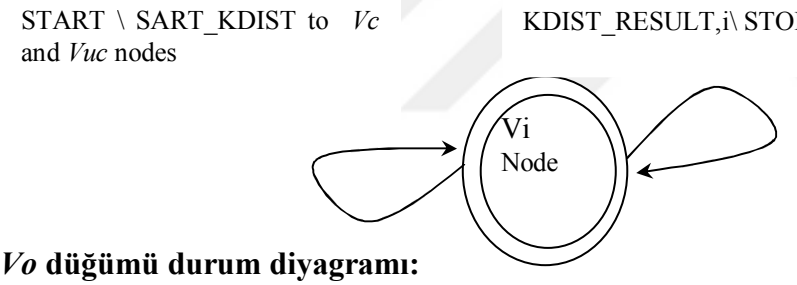
iii. KDIST_JOIN_ACK: Kök düğüme (taşırma mesajın sahibi) bilgi vermek amacıyla, *Vc* düğümleri tarafından den *Vuc* ya da *Vc* düğümlerine gönderilir.

iv. KDIST_JOIN_END: Kök düğüme (taşırma mesajın sahibi) taşırma işleminin bittiği bilgisini vermek amacıyla *Vc, Vuc* veya *Vo* düğümlerinden, *Vuc* ya da *Vc* düğümlerine gönderilir.

v. KDIST_RESULT: *k*-uzaklık komşuluk değerini iletmek amacıyla, *Vuc* ve *Vc* düğümlerinden, başlatıcı düğüme gönderilir.

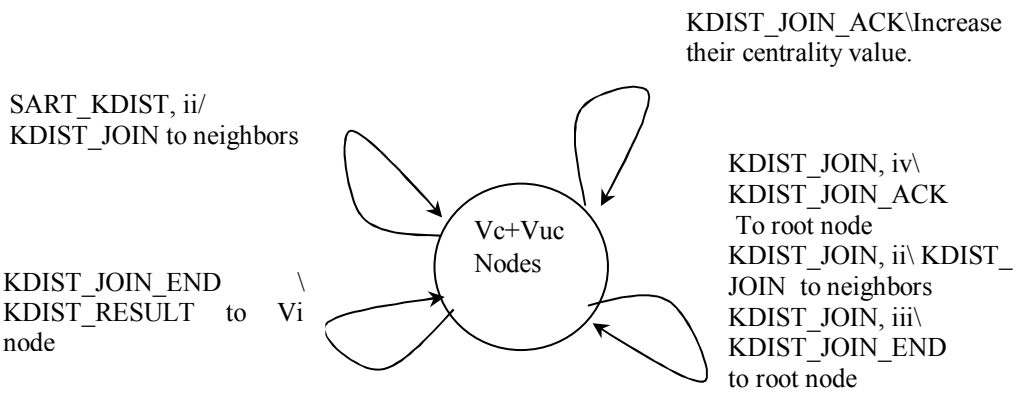
***k*-uzaklık merkezilik tabanlı kaynak atama algoritması durum diyagramı:**

***Vi* düğümü durum diyagramı:**



***Vo* düğümü durum diyagramı:**

***Vc* ve *Vuc* düğümlerin durum diyagramı:**



Şartlar:

i. Eğer KDIST_RESULT mesajı tüm *Vc* ve *Vuc* düğümlerden alındı ise aday listesi elde edilmiş demektir ve algoritma sonlanır.

ii. Eğer mesaj ilk defa alınıyorsa, *k*-uzaklık değeri aşılmadı ise alınan mesaj gönderici düğüm dışındaki tüm komşulara taşırma yöntemi ile iletilir.

iii. Eğer *k*-uzaklık değeri aşıldı ise ve daha önce gönderilmedi ise mesajın ait olduğu kök düğüme KDIST_JOIN_END mesajı iletilir.

iv. Eğer bir kök düğüme ait ilk defa KDIST_JOIN mesajı alındı ise ilgili kök düğüme KDIST_JOIN_ACK mesajı ile bilgi verilir.

v. Eğer KDIST_JOIN_END mesajı ilk defa alınıyorsa, başlatıcı düğüme KDIST_RESULT mesajı ile bilgi verilir.

Çizelge 3.2 KDCBC algoritması zaman ve mesaj karmaşıklıkları.

Mesaj Adı	**Zaman Karmaşıklığı**	**Mesaj Karmaşıklığı**
Start_Kdist	O(d)	Θ(Vc+Vuc)
Kdist_Join(Flooding)	O(k)	O(2 *E_k*(Vc+Vuc))
Kdist_join_Ack	O(k)	O((Vc+Vuc)*Vc)
Kdist_join_End	O(k)	O(2 *E_k*(Vc+Vuc))
Kdist_Result	O(d)	Θ(Vc+Vuc)
KDCBC Algoritması	O(d+k)	O((vc+Vuc)(2+ 4E_k +Vc))
	O(d)	**O(N_s + $N_s E_k$ + N_s) ≤ O(N + NE)**

Yapılan işlemlerin tamamına ait karmaşıklıkları bulabilmek için alt çizge belirleme algoritmasının karmaşıklığının bu sonuçlara ilave edilmesi gerekmektedir.

Çizelge 3.3 Çizelge 3.4 KDCBC ve SG belirleme algoritmalarının toplam zaman ve mesaj karmaşıklıkları.

TOPLAM	**Zaman Karmaşıklığı**	**Mesaj karmaşıklığı**
CCBC + KDCBC Algoritması	O(d) +O(d)	**O(E_s + N_s) + O(N_s E_k + N_s)**
	O(d)	**O(N_s + E_s + N_s E_k) ≤ O(N + E + NE)**

Burada *d* alt çizge çapını, E_s alt çizgedeki kenar sayısını, E_k *k* uzaklıkta bulunabilecek kenar sayısını, N_s alt çizgedeki düğüm sayısını temsil etmektedir.

Algoritma 3.2 *k*-Uzaklık Merkezilik Tabanlı Kaynak Atama Algoritması (KDCBC)

Amaç: *RSG* kümesi içindeki düğümlerin *k* kadar mesafede içinde kaç tane *Vc* düğüme ulaşabildiğini bulmak.

Girdiler: *Vc* ve *Vuc* düğümlerinden oluşan indirgenmiş alt çizge düğüm kümesi, *k* değeri ve başlatıcı düğüm numarası (İng. node id).

Çıktılar: *RSG* kümesindeki tüm *Vc* ve *Vuc* düğümlerine ait *k*-uzaklık merkezilik değerleri.

Sonlanma (İng. termination condition**):** Başlatıcı düğüm, *RSG* kümesindeki tüm *Vc* ve Vuc düğümlerine ait *k*-uzaklık merkezilik değerini aldığında algoritma sonlanır.

***Vi* node:**

1: kdist_Result_Counter ←*0*
2: **send** START_KDIST message to all RSG nodes
3: *Vi* node starts to **wait** KDIST_RESULT message of nodes
4: **Upon received** a KDIST_RESULT message:
5: **save** received centrality value to *k*-distance centrality list
6: **if** all of the subgraph nodes answered **then**
7: **sort** and **list** *k*-distance centrality list
8: **end of KDCBC** algorithm.
9: **end if**

***Other* nodes:**

1: **Upon received** a START_KDIST message :
2: **send** KDIST_JOIN message to all neighbors
3: start to **wait** KDIST_JOIN_END message from any node
4: **Upon received** a KDIST_JOIN message of related root node :
5: **if** receiver is a *Vc* node and first message of that root **then**
6: **send** a KDIST_JOIN_ACK message to message root
7: **if** distance of message is not greater than the *k*-distance **then**
8: **add** their distance to message distance
9: **send** KDIST_JOIN message to all neighbors
10: **else**
11: **send** a KDIST_JOIN_END message to message root
12: **end if**
13: **Upon received** a KDIST_JOIN_ACK message :
14: Increment their k-distance centrality value

15: **Upon received** a first KDIST_JOIN_END message :
16: **if** it is received first time **then**
17: **send** a KDIST_RESULT message to Vi node with centrality value

---

3.4 Aradalılık Merkezilik Tabanlı Kaynak Atama Algoritması (İng. Betweenness Centrality Based Candidate Generation Algorithm BCBC)

BCBC merkezilik algoritması geçirgenlik sınıfının bir üyesidir. Bu tür merkezilikler genellikle çizgelerde köprü konumlarını ve tıkanıklık olabilecek düğümleri belirlemek ya da kümeleme (İng. clustering) yapmak gibi birçok amaçla kullanılabilmektedir. BCBC algoritmasının amacı ise alt çizge içinde bulunan düğümler arasındaki köprü konumlarını belirlemektir. Köprü konumundaki düğümlerden, kaynak atama işlemi için aday listesi oluşturulacaktır. Aday listesinde bulunan en yüksek merkezilik değerine sahip düğümler, sorgu işleme birimi olarak seçilecektir. Böylece veri kaynakları ile sorgu işleme birimi arasındaki veri trafiğinin en aza indirilmesi amaçlanmaktadır. Sınırlı aradalılık merkezilik değerlerini hesaplamak için tüm düğümlerden tüm düğümlere olan kısa yolların belirlenmesi yerine, *Vc* düğümlerinden *Vc* düğümlerine ve *Vc* düğümlerinden *Vuc* düğümlerine olan tüm kısa yolarlın belirlenmesi yeterlidir. Bunun nedeni Lemma-1.1 de açıklanmıştır. Bu algoritma çalıştırılmadan önce, alt çizge belirleme algoritması (CCBRA) çalıştırılmalı ve indirgenmiş alt çizge kümesi belirlenmelidir. BCBC algoritmasının karmaşıklıkları diğerlerine göre daha yüksek olduğu için aday belirleme maliyetini düşürmek amacıyla *RSG* kullanılmıştır. Algoritmanın çalışması sonucu, *RSG* kümesindeki her düğümün aradalılık merkezilik değerleri sıralanarak aday listesi oluşturulmaktadır. Bu doğrultuda hazırlanan algoritma aşağıdaki gibidir.

Mesaj Tipleri:

i. **BTW_START:** Başlatıcı *(Vi)* düğümden, *Vc* ve *Vuc* düğümlerine sınırlı aradalılık merkezilik işlemini başlatmaları için gönderilen mesajdır.

ii. **BTW_END:** *Vc* ve *Vuc* düğümlerinden, başlatıcı *(Vi)* düğüme sınırlı aradalılık merkezilik işleminin bittiğini bildirmek için gönderilen mesajdır.

iii. **BTW_PROBE:** Taşırma (İng. flooding) yoluyla, en kısa yolların belirlenmesi amacıyla, *Vc, Vuc* ya da *Vo* düğümleri tarafından, *Vc, Vuc* ya da *Vo* (komşu) düğümlerine gönderilen mesajdır.

iv. BTW_RESULT: Komşu düğümlerden ebeveyn (İng. parent) düğümlere gönderilen mesajdır. Bu mesajla her düğüme ait aradalılık değeri oluşmaya başlar (İng. convergecasting).

v. BTW_V_REQ: Başlatıcı *(Vi)* düğümden *Vc* ve *Vuc* düğümlerine sınırlı aradalılık merkezilik değerlerini göndermeleri için gönderilen mesajdır.

vi. BTW_VALUE: *Vc* ve *Vuc* düğümlerinin başlatıcı *(Vi)* düğüme, sınırlı aradalılık merkezilik değerlerini bildirmek için gönderdikleri mesajdır.

Çizelge 3.4 BCBC algoritması zaman ve mesaj karmaşıklıkları.

Mesaj Adı	**Zaman Karmaşıklığı**	**Mesaj Karmaşıklığı**
Btw_Start	O(d)	Θ(Vc+Vuc)
Btw_End	O(d)	Θ(Vc+Vuc)
Btw_Probe	O(d)	O(2*Es*(Vc+Vuc))
Btw_Result	O(d)	O((Vc+Vuc)*hop)
Btw_V_Request	O(d)	Θ(Vc+Vuc)
Btw_Value	O(d)	Θ(Vc+Vuc)
BCBC Algoritması	O(d)	O(4*(Vc+Vuc)+ hop*(Vc+Vuc)+ 2*Es*(Vc+Vuc))
	O(d)	**O((Vc+Vuc)*(hop+2Es+4))≤O(N^2 + NE)**

Yapılan işlemlerin tamamına ait karmaşıklıkları bulabilmek için alt çizge belirleme algoritmasının karmaşıklığının bu sonuçlara ilave edilmesi gerekmektedir.

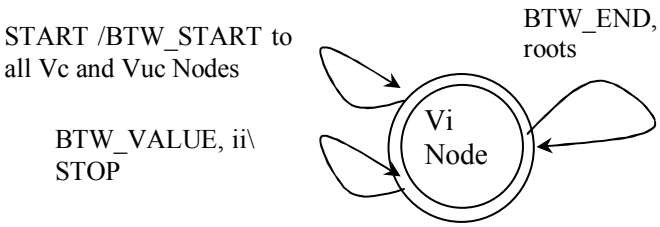
Çizelge 3.5 BCBC ve SG belirleme algoritmalarının zaman ve mesaj karmaşıklıkları.

TOPLAM	**Zaman Karmaşıklığı**	**Mesaj karmaşıklığı**
CCBC + BCBC Algoritması	O(d) +O(d)	O(E_s + N_s) + O(N_s^2 + N_sE_s)
	O(d)	**O(N_s + E_s + N_s^2 + N_sE_s)≤O(N + E + N^2+ NE)**

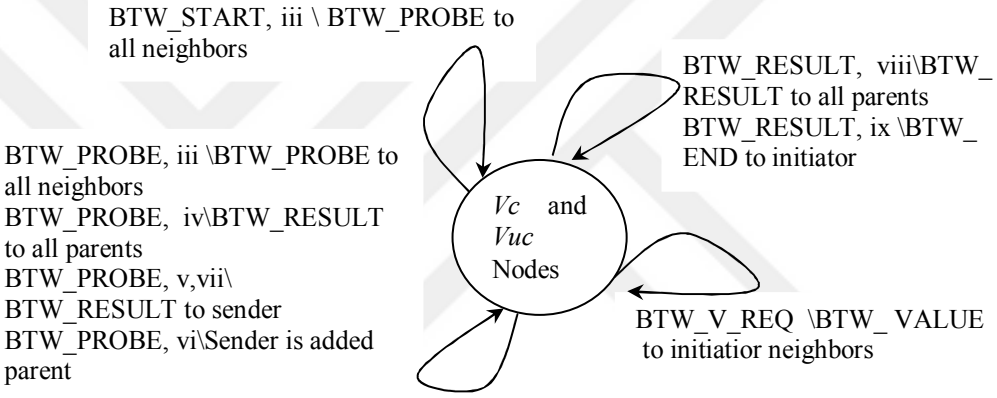
Burada *d* alt çizge çapını, E_s alt çizgedeki kenar sayısını, *hop* alt çizgedeki atlama sayısını, N_s alt çizgedeki düğüm sayısını temsil etmektedir.

Aradalılık merkezilik tabanlı kaynak atama algoritması durum diyagramı:

***Vi* düğümü durum diyagramı:**



***Vc* ve *Vuc* düğümleri durum diyagramı:**



Şartlar:

i. Eğer *Vc* ve *Vuc* düğümlerinin hepsi BTW_END mesajı gönderdi ise

ii. Eğer *Vc* ve *Vuc* düğümlerin hepsi BTW_VALUE mesajı gönderdi ise

iii. Eğer ilgili kök düğüme ait ilk defa BTW_PROBE mesajı alındı ise birden fazla komşusu varsa ve uzaklık *Vc* çapı aşmadı ise

iv. Eğer ilgili kök düğüme ait ilk defa BTW_PROBE mesajı alındı, düğüm bir *Vc* ve çizgede yaprak (İng. leaf) konumunda ise merkezilik değerini 1 yaparak

v. Eğer ilgili kök düğüme ait ilk defa BTW_PROBE mesajı alındı, düğüm *Vc* değilken, düğüm yaprak konumunda veya uzaklık *Vc* çapını aştı ise merkezilik değerini 0 yaparak

vi. Eğer ilgili kök düğüme ait daha önce BTW_PROBE mesajı alındı ve gelen mesajın uzaklığı benim önceki ebeveynime ait mesajın uzaklığına eşit ya da küçükse ise

vii. Eğer ilgili kök düğüme ait daha önce BTW_PROBE mesajı alındı ve gelen mesajın uzaklığı benim önceki ebeveynime ait mesajın uzaklığından büyük ise merkezilik değerini 0 yaparak

viii. Eğer tüm komşularım cevap verdi ve düğüm bu mesajın kök düğümü değilse

ix. Eğer tüm komşularım cevap verdi ve düğüm bu mesajın kök düğümü ise.

Algoritma 3.3 Aradalılık Merkezilik Tabanlı Kaynak atama Algoritması (BCBC).

Amaç: RSG kümesi içindeki düğümlerin, sınırlı aradalılık merkezilik değerlerini hesaplamak.

Girdiler: *Vc* ve *Vuc* düğümlerinden oluşan alt çizge düğüm seti, alt çizge çapı, başlatıcı düğüm numarası (İng. node id).

Çıktılar: *Vc* ve *Vuc* düğümlerine ait sınırlı aradalılık merkezilik değerleri.

Sonlanma (İng. termination condition): Başlatıcı düğüm, *RSG* kümesindeki tüm *Vc* ve *Vuc* düğümlerine ait aradalılık merkezilik değerlerini aldığında algoritma sonlanır.

***Vi* node:**

1: **send** BTW_START message to all *Vc* and *Vuc* nodes in RSG
2: **set** Received_End_Message_Counter← number of *Vc* and *Vuc* nodes
3: starts to **wait** BTW_END message of nodes
4: **Upon received** a BTW_END message :
5: **decrement** Received_End_Message counter
6: **if** Received_End_Message_Counter is zero **then**
7: **send** BTW_V_REQ message to all *Vc* and *Vuc* nodes
8: **set** Received_Value_Message_Counter ← number of *Vc* and *Vuc* nodes
9: **end if**
10: start to **wait** BTW_VALUE message of *Vc* and *Vuc* nodes
11: **Upon received** a BTW_VALUE message :
12: **decrement** Received_Value_Message_Counter
13: **save** sender centrality value
14: **if** Received_Value_Message_Counter is zero **then**
15: **sort and list** centrality of nodes
16: **end** of **BCBC** algorithm
17: **end if**

***Other* nodes:**

```
1:  Upon received a BTW_START message:
2:      if distance of message is not greater than the Vc diam then
3:                          send BTW_PROBE message to all neighbors
4:      start to wait BTW_RESULT/ BTW_REJECT message of neighbors
5:  Upon received BTW_ PROBE message of related root node :
6:    if first BTW_ PROBE message of related root node then
7:            set Btw_Centrality ← 0
8:          set sender as my parent
9:          if I am a Vc node and (I have no neighbor or distance of message is greater than the
    Vc diam) then
10:           set Btw_Centrality ← 1
11:           send a BTW_RESULT message to parent with my centrality
12:           end if
13:         if I am not a Vc node and (I have no neighbor or distance of message is greater than
    the Vc diam) then
14:               send a BTW_RESULT message to parent with my centrality
15:           if I have more than one neighbor then
16:            if distance of message is not greater than the Vc diam then
17:               send BTW_PROBE message to all neighbors except sender and root
18:           end if
19:           else
20:         if distance of message  is small or equal to first parent distance then
21:               add sender node to parent list
22:           else
23:         send a BTW_RESULT message to sender with zero centrality
24:           end if
25:         end if
26:  Upon received a BTW_RESULT message from a neighbor:
27:   if I am root of messge then
28:         add received centrality value to my centrality
29:         if all chiIdren sent result message then
30:           send a BTW_END message to initiator node
31:         else
32:         add received centrality value to my related root centrality
33:         if all chiIdren of related root sent result message then
34:           send a BTW_RESULT message to related root node parent
35:           end if
36:  Upon received a BTW_V_REQ message from Vi node:
```

37: **send** a BTW_VALUE message to *Vi* with centrality value

3.5 Kıyaslama Algoritmaları

Önerilen algoritmaların başarılarını test etmek amacıyla, daha önceki çalışmalardan faydalanarak iki adet kıyas algoritması kodlanmıştır. Literatür taramasında pek çok çalışmada kullanılan maliyet modelinde, ağ kıstası olarak sadece kanal genişliğinin (İng. bandwidth) kullanıldığı gözlenmiştir. Bu tür çalışmaları temsil etmesi amacıyla, Kanal Genişliği Tabanlı Kıyas Algoritması (İng. BandWidth Based Compare Algorithm BWBCA) olarak adlandırılan algoritma kodlanmıştır. Çünkü bu tez çalışmasının amaçları doğrultusunda hazır bir algoritma bulunamamıştır. Bildiğimiz kadarı ile bu tez çalışmasına en yakın çalışma olan, Çokuslu ve arkadaşlarının çalışmasında ise kaynak seçiminde ağ kıstası olarak veri kaynakları ve sorgu birimleri arasındaki veri iletim süresi kullanılmıştır. Bu nedenle ikinci kıyas algoritması olarak, Yakınlık Tabanlı Aday Listesi Oluşturma Algoritması (İng. Proximity Based Candidate list Generation algorithm PBCGA) adını verdikleri algoritma kullanılmıştır.

Her iki algoritmada; daha önce bahsedilen kabullenmeler doğrultusunda, önerilen algoritmalar ile benzer özellikler kullanılarak, aynı benzetim ortamında kodlanmıştır. Aynı veriler ile aynı şartlarda test senaryoları gerçekleştirilmiştir. Bu sayede kıyaslamanın adil olması sağlanmıştır.

3.5.1 Yakınlık Tabanlı Aday Listesi Oluşturma Algoritması (İng. Proximity Based Candidate List Generator PBCGA)

Bu algoritma, önerilen algoritmaların; zaman, mesaj karmaşıklıklarını ve daha önemlisi belirledikleri adayların sorgu işlemindeki performanslarını daha önceki çalışmalarla kıyaslamak için kullanılacak ilk kıyaslama algoritmasıdır. PBCGA algoritması, kabullenmeler ve daha önce yapılan çalışmalar dikkate alınarak kodlanmıştır. Bildiğimiz kadarı ile ilingesel özellikleri dikkate alarak geliştirilen sadece bir kaynak atama algoritması bulunmaktadır ve bu çalışma Çokuslu ve arkadaşlarına aittir (Çokuslu et al. 2012b). PBCGA algoritmasında adaylar tüm veri kaynaklarına en hızlı ulaşan düğümlerden seçilmektedir. Bu tez çalışmasında önerilen yakınlık merkezilik tabanlı kaynak atama algoritması (CCBC) ile çok büyük benzerlik göstermektedir. CCBC algoritmasında alt çizge üzerinde sınırlı yakınlık merkezilik tabanlı bir yaklaşım uygulanmış ve düğümler

arası uzaklık dikkate alınırken, PBCGA'da ise düğümlerin birbirlerine olan erişim sürelerini dikkate almışlardır.

PBCGA algoritması, tüm veri kaynaklarından başlatılan ve veri kaynaklarının çapı kadar yayılacak taşırma mesajlarını kullanmıştır. Veri kaynaklarından gönderilen taşırma mesajlarının tümünü alan bir düğüm, bunu başlatıcı düğüme bir mesaj ile ileterek aday olduğunu bildirmektedir. Başlatıcı düğüm, düğümlerden gelen mesajları geliş zamanının göre sıralamakta ve ilk gelen mesajın sahibini tüm veri kaynaklarına en yakın düğüm olarak kabul etmektedir. PBCGA algoritması, başlatıcı düğüme en uzak olan iki düğümden herhangi birinden mesaj aldığında ise sonlanmaktadır.

Bu tez çalışmasındaki kabullerimize göre başlatıcı, *Vc* düğümlerinden herhangi biri olabilir ve ona en uzak iki düğüm bilgisi ise kabullerimiz arasında bulunmamaktadır. Oysa PBCGA algoritmasında sonlanma işlemi, bu en uzak iki düğümün birisinden başlatıcı düğüme gelecek mesajla sağlanmaktadır. Bu nedenle önerilen diğer algoritmalarla eşit şartlarda çalıştırmak amacıyla, PBCGA algoritmasının sonlanması, başlatıcı düğüme tüm *Vc* düğümlerinden cevap geldiğinde olcak şekilde değiştirilmiştir.

Çizelge 3.6 PBCGA algoritmasının zaman ve mesaj karmaşıklıkları.

Mesaj Adı	**Zaman Karmaşıklığı**	**Mesaj Karmaşıklığı**
Start_Vc_Proximity	O(d)	$\Theta(Vc)$
Vc_Probe (Flooding)	O(d)	$O(4*Vc*E_s)$
Vx_ Proximity	O(d)	$\Theta(N_s)$
PBCGA Algoritması	O(d)	$O(Vc + 4VcE_s + N_s)$
Sonuç	**O(d)**	$\mathbf{O(E_s + N_s) \leq O(N + E)}$

Burada *d* alt çizge çapını, E_s alt çizgedeki kenar sayısını, , N_s alt çizgedeki düğüm sayısını temsil etmektedir.

Bu algoritmanın aday belirleme süresi (çalışma süresi), kullandığı mesaj sayısı ve belirlediği adaylar önerilen algoritmaların sonuçları ile kıyaslanacaktır.

3.5.2 Kanal Genişliği Tabanlı Kıyaslama Algoritması (İng. Bandwidth Based Compare BWBCA)

Bu algoritma önerilen algoritmaların karmaşıklıklarını ve daha önemlisi belirledikleri adayların sorgu işlemindeki performanslarını, daha önceki çalışmalarla kıyaslamak için kullanılacak ikinci kıyas algoritmasıdır. BWBCA algoritması da daha önce yapılan kabullenmeler ve tanımlanan özellikler dikkate alınarak kodlanmıştır. Bildiğimiz kadarı ile ilingesel özellikleri dikkate alarak geliştirilen sadece bir kaynak atama algoritması bulunmamaktadır (Çokuslu et al. 2012b). Bu nedenle geçmişteki diğer çalışmaların da temsil edilebilmesi amacıyla ikinci bir kıyaslama algoritması kodlanmıştır. Daha önceki çalışmalarda kullanılan maliyet modelleri incelendiğinde ağ yapısına yönelik kullanılan yegâne parametre kanal genişliğidir. Birkaç çalışmada bir düğümün ortalama kanal genişliği hesaplanarak maliyet modeline eklendiği görülmüştür (Gounaris et al. 2006a). Bu çalışmalara dayanarak kıyas algoritması, ağdaki tüm düğümleri ortalama kanal genişliğine göre sıralayacak şekilde hazırlanmıştır. Bu tez çalışması düğümlerin ilingesel özelliklerini öne çıkarmak amacıyla hazırlandığı için aday seçimindeki diğer tüm kıstasların (kanal genişliği ve ilingesel özellikler hariç) tüm düğümler için eşit olduğu kabul edilmiştir. Bu nedenle kıyas algoritması, tüm çizge içinde ortalama kanal genişliği en büyük olan düğümü en iyi aday olarak seçecektir. Bir düğüme ait ortalama kanal genişliği ise aşağıdaki denklemde olduğu gibi hesaplanmaktadır.

$$\text{Ortalama Kanal Genişliği(v)} = \frac{\sum Kenar\ Kanal\ genişliği\ (v,u)}{\sum Kenar(v,u)} \quad \text{u,v } \in \text{ V ve } u, v \text{ nin komşusudur.} \qquad (3.3)$$

Bu algoritmanın aday belirleme süresi (çalışma süresi), kullandığı mesaj sayısı ve belirlediği adaylar önerilen algoritmaların sonuçları ile kıyaslanacaktır.

Mesaj Tipleri:

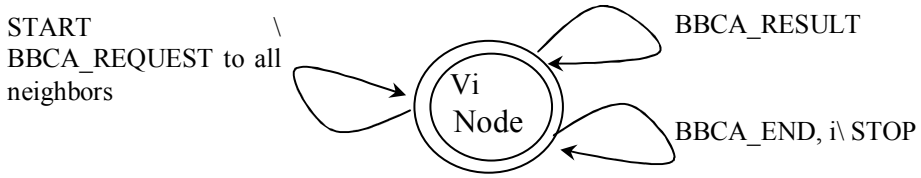
i. BBCA_REQUEST: Çizgedeki tüm düğümlere, taşırma yolu ile kendilerine ait ortalama kanal genişliği değerlerini göndermeleri için *Vi* düğümü tarafından gönderilir.

ii. BBCA_RESULT: *Vi* düğümüne ortalama kanal genişliği bilgisi vermek amacıyla gönderilen bilgi mesajıdır. Gönderme işlemi, taşırma anında oluşan önce enine arama ağacı (İng. Breadth First Search *BFS*) üzerinden yapılır

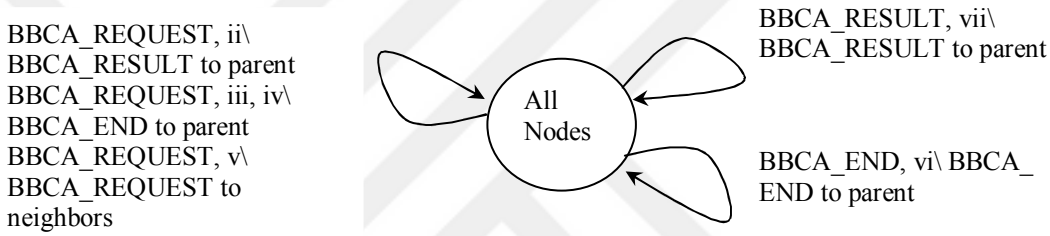
iii. BBCA_END: BBCA_REQUEST mesajı gönderen düğümlere, cevap vermek amacıyla gönderilen mesajdır. Normalde oluşan *BFS* ağacının en altından başlar ve başlatıcı düğüme kadar ulaşır.

Kanal genişliği tabanlı kıyas algoritmasının durum diyagramı:

***Vi* düğümü durum diyagramı:**



Diğer düğümlerin durum diyagramı:



Şartlar:

i. Eğer tüm komşulardan BBCA_END mesajı alındı ise

ii. Eğer BBCA_REQUEST mesajı ilk defa alınıyorsa

iii. Eğer düğümün göndericiden başka komşusu yoksa yani yaprak konumunda ise

iv. Eğer BBCA_REQUEST mesajı daha önce alındı ise

v. Eğer BBCA_REQUEST mesajı ilk defa alınıyor ve birden fazla komşusu var ise

vi. Eğer kendisi Vi düğümü değil ise ve diğer tüm çocuklarımdan BBCA_END mesajı alındı ise

vii. Eğer kendisi Vi düğümü değil ise

Çizelge 3.7 Kanal genişliği tabanlı kıyas algoritmasını zaman ve mesaj karmaşıklıkları.

Mesaj Adı	Zaman Karmaşıklığı	Mesaj Karmaşıklığı
BBCA_Request	O(d)	Θ(2E)
BBCA_Result	O(d)	Θ(N*hop)
BBCA_End	O(d)	Θ(E)
BWBCA Algoritması	O(d)	O(3E+N*hop)
	O(d)	**O(N+E)**

Burada *d* ana çizge çapını, *hop* maksimum atlama sayısını, *E* ana çizge kenar sayısını ve *N* ise düğüm sayısını temsil etmektedir.

Algoritma 3.4 Kanal genişliği tabanlı kıyas Algoritması (BBCA).

Amaç: Çizgedeki düğümlerin kanal genişliklerini büyükten küçüğe sıralamak
Girdiler: Ana çizge düğüm seti, başlatıcı düğümün numarası (*Vi*) (İng. node id).
Çıktı: Büyükten küçüğe sıralı, düğümlere ait ortalama kanal genişliği değerleri.
Sonlanma (İng. termiantion)**:** Başlatıcı düğüm tüm komşularından BBCA_END mesajı aldığında sağlanmaktadır.

***Vi* node:**

1: **add** BWList ← my average BW
2: **set** Wait_Msg_Counter ← number of neighbor
3: **send** BBCA_REQUEST message to neighbors
4: start to **wait** BBCA_RESULT or BBCA_END message of nodes
5: **Upon received** a BBCA_RESULT message
6: **add** BWList ← average bandwidth value of sender node
7: **Upon received** a BBCA_END message
8: **decrement** Wait_Msg_Counter
9: **if** Wait_Msg_Counter is zero **then**
10: **sort** and **list** received average bandwidth value of nodes
11: **end of** BWBCA algorithm.
12: **end if**

Other nodes:

```
1:  Upon  received a BBCA_REQUEST
2:              If it is received first BBCA_REQUEST message  then
3:              Parent ← sender
4:              calculate average bandwidth
5:              set Wait_Msg_Counter ← number of neighbor
6:              send BBCA_RESULT message to parent with average bandwidth
7:              if  Wait_Msg_Counter -1 is zero then
8:                  send BBCA_END message to parent node
9:                  else
10:                 send BBCA_REQUEST message to neighbors except sender
11:                 end if
12:             else
13:         send BBCA_END message to sender node
14:             end if
15:  Upon  received a BBCA_RESULT message
16:             send BBCA_ RESULT message to parent node
17:  Upon  received a BBCA_END message
18:             decrement Wait_Msg_Counter
19:         if  Wait_Msg_Counter is one then
20:             send BBCA_END message to parent node
```

3.6 Sorgu Birleştirme Benzetim Algoritması (İng. Query Join Simulation QJSA)

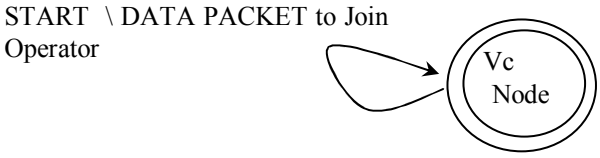
Sorgu birleştirme benzetim algoritması, önerilen algoritmalar ve kıyas algoritmaları tarafından belirlenen adayların performanslarını karşılaştırmak amacıyla kullanılacaktır. Bilindiği gibi istenilen sorgunun yapısına ve transfer edilecek verinin büyüklüğüne bağlı olarak, bir sorgu işlemi son derece karmaşık ve yoğun bir işlem gerektirebilir. Kısacası sorgu işleme performansını etkileyen birçok parametre bulunmaktadır ve gerçek bir sorguda bu parametrelerin birçoğu dikkate alınarak kaynak atama işlemi yapılmaktadır. Bu çalışmanın amacı ilingesel parametrelerin etkisini araştırmak ve bu yaklaşımla iletişim maliyetini düşürecek çözümler üretmek olduğundan, diğer parametrelerin tüm düğümler için eşit olduğu kabul edilmiştir. Bu nedenle daha önce tanımlanan kabuller de dikkate alınarak, seçilen adayın ilingesel özelliklerinin etkinliğini ortaya çıkartacak bir benzetim ortamı oluşturulmaya çalışılmıştır. Benzetim algoritmasını basitleştirmek ve anlaşılır kılmak adına sistemde bir tane veri birleştirme operatörü (İng. single join operator) olduğu, bir veri kaynağındaki verinin bir

başka veri kaynağında bulunmadığı, her bir veri kaynağından aynı miktarda veri transfer edileceği ve her bir veri kaynağının tablonun yatay olarak bölünmüş bir parçasını tuttuğu kabul edilmiştir.

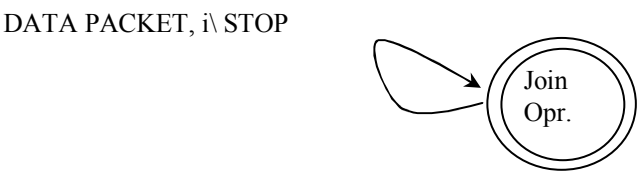
Benzetim algoritması, ilgili çizgede daha önce tanımlanmış olan veri kaynaklarından (*Vc*) birleştirme operatörü/sorgu işleme birimi olarak belirlenen aday düğüme yüklü miktarda veri aktarımını yapacak şeklinde tasarlanmıştır. Her aday için aktarılan veri miktarı sabit olacağı için veri aktarım süresi test edilen adayın etkinliğini gösterecektir. Bu yaklaşım, adayların etkinliklerini ortaya çıkarmak için oldukça basitleştirilmiş bir sorgu işleme benzetimidir ve bir benzeri Çokuslu ve arkadaşlarınca da kullanılmıştır (Çokuslu et al. 2012b). Bu aşamada benzetim ortamının, belirlenen amaçlar doğrultusunda gerçek hayat şartlarına uygun olması için özen gösterilmiştir. Bunun için algoritma daha önce belirtilen kabuller de dikkate alarak, *TCP* ve *NS3* (Network Simulator 3) evrensel ağırlıklı *IP* yönlendirme (İng. Global Weighted *IP* Routing) protokollerini kullanmaktadır. Adayların arasındaki farkı daha net ortaya çıkarabilmek için büyük miktarda veri transferi yapılması amaçlanmıştır. Ayrıca testlerin tarafsızlığını ve akademik çevrelerce kabul edilebilirliğini arttırmak için benzetim ortamı olarak *NS3* ağ benzeştiricisi kullanılmıştır. Veri transfer işlemi için *NS3* geliştiricileri tarafından sağlanan "*TCP* büyük veri transfer" (İng. *TCP* Large Data Transfer) uygulaması kullanılmıştır.

Sorgu Birleştirme Benzetim Algoritmasının durum diyagramı:

***Vc* düğümünün durum diyagramı:**



Birleştirme operatörü (JOP), düğümünün durum diyagramı:



Şart:

i. Eğer toplam veri transferi sağlandı ise.

Çizelge 3.8 Sorgu Birleştirme Benzetim Algoritmasının zaman ve mesaj karmaşıklıkları

Mesaj Adı	Zaman Karmaşıklığı	Mesaj Karmaşıklığı
Data Packet	O(Data/Packet_Size*d)	O(Data/Packet_Size*Vc)
QJSA Algoritması	**O(Data/Packet_Size*d)**	**O(Data/Packet_Size*Vc)**

Burada *d* alt çizge çapını, Vc veri kaynağı sayısını, Data bir veri kaynağından transfer edilecek veri miktarını ve *Packet_Size* ise *TCP* protokolünün o anki şartlarda gönderebildiği en büyük veri paketi büyüklüğünü temsil etmektedir.

Algoritma 3.5 Sorgu Birleştirme Benzetim Algoritması (QJSA).

Amaç: Veri kaynaklarından sorgu birleştirme operatörüne iletilen veri transfer süresini ölçmek.

Girdiler: Ana çizge düğüm seti, birleştirme Operatörü, *Vc* düğüm seti, bir *Vc* tarafından transfer edilecek veri miktarı.

Çıktı: Veri transfer süresi.

Sonlanma (İng. termination): Birleştirme operatörü olarak görev yapan düğüm, aktarılması gereken toplam veriyi aldığında program sonlanmaktadır.

Each data resource:

1: **set** Number of send data block
2: **while** (Number of send data block is not zero)
3: **send** a data block to join operator
4: **decrement** Number of send data block
5: **end while**

join operator:

1: **save** simulation Start_Time
2: **Upon received** a data block
3: **increment** Number of received data block
4: **if** Number of received data block = To be received total data block **then**
5: **save** simulation Stop_Time
6: **print** Stop_Time – Start_Time
7: **end** QJSA algorithm

8: **end if**

---

4. BENZETİMLER

Bu bölümde, önerilen algoritmalar ve kodlanan kıyas algoritmaları, aşağıda açıklanan senaryolar doğrultusunda çalıştırılmış ve elde edilen sonuçlarının değerlendirmeleri yapılmıştır.

Yapılan araştırmalar sonucunda, istenilen özelliklerde gerçek çizge verisi bulunamamıştır. Bu nedenle *BRITE* isimli ilinge üreteci kullanılarak 100, 250, 500, 750, 1000, 1250 ve 1500 düğümlük çizge verileri oluşturulmuştur. Bu çizgeler daha önce belirtilen kabullenmeler doğrultusundadır ve internet ağ özelliklerini taşımaktadır. Çizelge 4.1 de kullanılan her bir çizgeye ait temel özellikler verilmiştir. Bu çalışmada kısa yolları belirlemek için düğümler arası mesaj iletim süresi kullanılmıştır, sorgu benzetiminde ise ağırlıklı IP yönlendirme protokolü kullanılmıştır. Bu nedenle aynı ağırlığa sahip kenarların mesajları aynı sürede iletebilmesi için kenar ağılıkları ile kenar kanal genişliği arasında bir ilişki kurulmuştur. Bu ilişki Denklem 3.2 de tanımlanmıştır. Düğümlere ait en büyük kanal genişliği 1Gbps olarak alınmış ve bu denklem ile kanal genişliklerinin 1Gbps ile 100 Mbps arasında olması sağlanmıştır. Her çizge içinden, veri kaynaklarınını (*Vc*) temsil etmek amacıyla 20 adet düğümden oluşan *VC* kümeleri oluşturulmuştur. Bu kümeler, belirli bir çapa sahiptir ve içindeki *Vc* düğümler rastgele seçilmiştir. Veri birleştirme benzetim aşamasında ise her bir veri kaynağından sorgu birleştirme operatörüne 100 MByte veri transferi yapılmıştır. Böylece, toplam 2 GByte lık bir veri transferi sağlanmıştır.

Çizelge 4.1 Benzetim aşamasında kullanılan çizgelerin temel özellikleri.

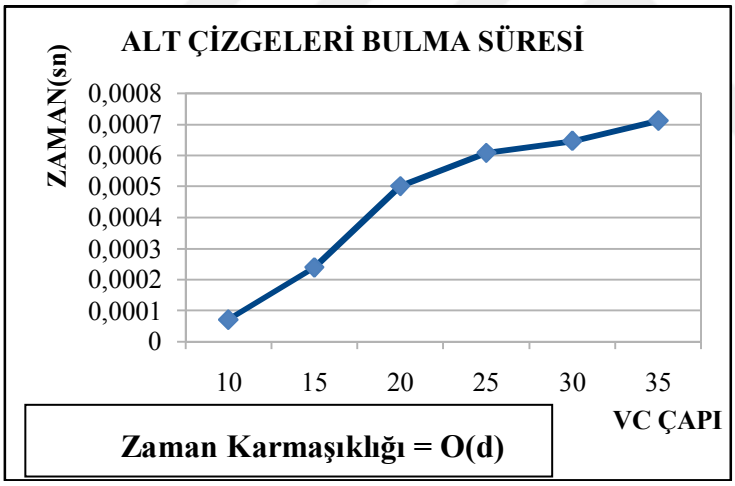
Düğüm sayısı	Kenar sayısı	En büyük derece	Ortalama derece	Ana çizge çapı (ağırlıklı)	Ana çizge çapı (atlama)	Vc düğüm sayısı	Vc düğüm çapı (ağırlıklı)
100	**200**	**14**	**4**	**24**	**6**	**20**	**20**
250	**497**	**58**	**3**	**36**	**6**	**20**	**20**
500	**997**	**54**	**3**	**39**	**7**	**20**	**20**
750	**1497**	**48**	**3**	**43**	**7**	**20**	**20**
1000	**1997**	**81**	**3**	**43**	**7**	**20**	**20**
1250	**2497**	**106**	**3**	**43**	**7**	**20**	**20**
1500	**2997**	**103**	**3**	**48**	**8**	**20**	**20**

4.1 Alt Çizge Belirleme Algoritmasının Benzetim Sonuçları

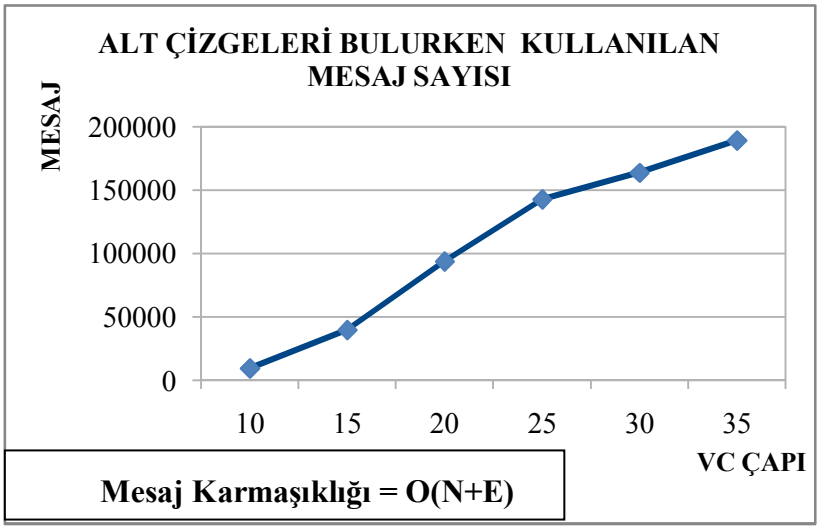
4.1.1 Çizge boyutu sabitken, *Vc* çap değişimine karşı elde edilen sonuçlar

Bu aşamada CCBC algoritmasının, çizge boyutu sabitken farklı *Vc* çaplarına karşı davranışı benzetim ortamında test edilmiştir. Doğal olarak *Vc* çapının artışı ile alt çizge boyutunda da bir artış beklenmektedir. Dolayısı ile *Vc* çapının algoritmanın performansını ve alt çizge boyutuna etkisini gözlemlemek amacıyla bu test yapılmıştır.

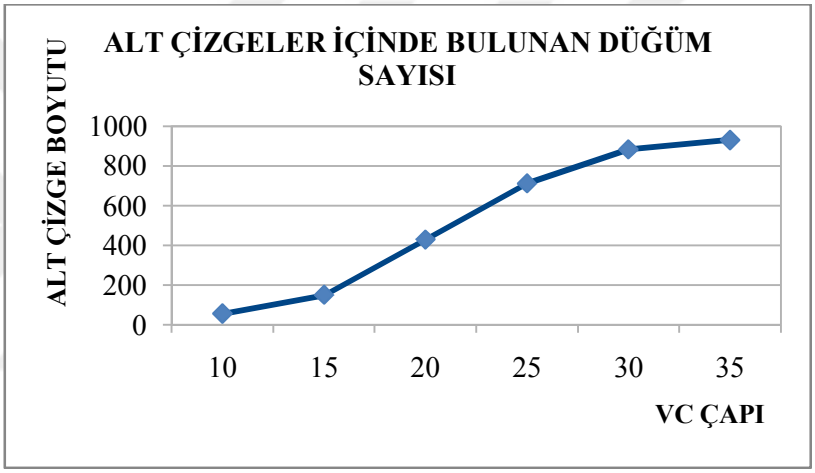
Test aşamasında ana çizge boyutu sabit tutulmuş, 1000 düğümlü, 1997 kenarlı ve çapı 43 birim olan bir çizge kullanılmıştır. *Vc* çapı olarak 10 dan 35 e kadar 6 farklı değer alınmış ve her *Vc* çapı için 5 farklı *VC* kümesi üzerinde deneme yapılmıştır. Elde edilen sonuçların ortalamaları alınarak aşağıdaki grafikler elde edilmiştir. Mesaj sayısı içine, her mesaj için en az bir kere gönderilmiş olan bilgilendirme (İng. acknowledgement) mesajları da dâhildir.



Şekil 4.1 Çizge çapı değişirken *SG* belirleme süresi.



Şekil 4.2 Çizge çapı değişirken *SG* belirlemek için gereken mesaj sayısı.



Şekil 4.3 Çizge çapı değişirken belirlenen *SG* boyutu.

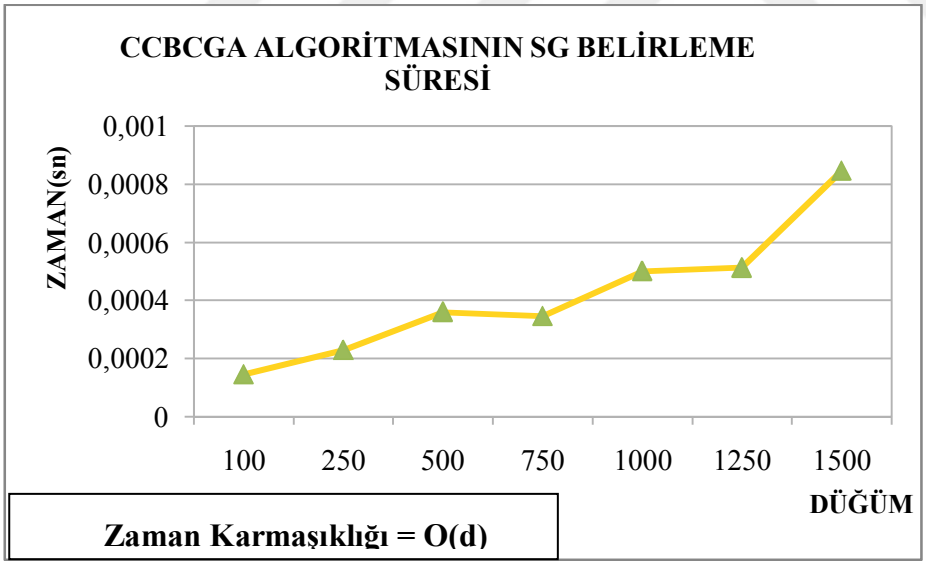
Benzetim sonuçları ve karmaşıklık değerlerine bakılarak bu algoritmanın ölçeklenebilir olduğu söylenebilir. *Vc* çap arttıkça zaman, mesaj ve alt çizge boyutu artmaktadır. Buradan alt çizge boyutunun, *Vc* çaptan daha çok *Vc* çapın ana çizge çapıyla olan oranına bağlı olduğu gözlenmektedir. Yani *Vc* çapı/Çizge çapı ne kadar büyükse alt çizge boyutu da o kadar büyük olacaktır. Grafikler *Vc* çapı 30 u geçtikten sonra yataylaşma eğilimi göstermektedir. Bunun nedeni bu noktadan sonra alt çizge boyutunun, ana çizge boyutuna çok yaklaşması olarak yorumlanmıştır.

4.1.2 *Vc* Çapı sabitken, çizge boyutu değişiminin alt çizge boyutuna olan etkisi

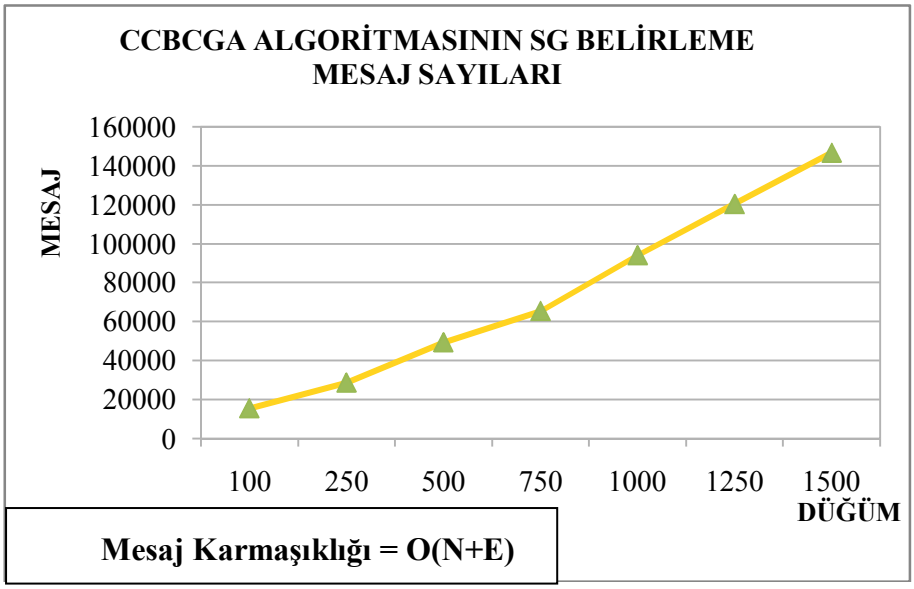
Bu bölümde alt çizge belirleme algoritması kullanımının, ana çizge boyutundan bağımsızlık sağlayıp sağlamadığı incelenmiştir. Bu amaçla

algoritmanın, *Vc* çapı sabit bir değere sahipken, çizge boyutu değişimine karşı davranışı benzetim ortamında test edilmiştir. Alt çizge boyutunu etkileyen temel faktörler; *i. Vc* çapı, *ii. Vc* çapının ana çizge çapına oranı, *iii.* Genel çizge yoğunluğu ve *iv. Vc* düğümlerin bulunduğu bölgenin yoğunluğu olarak sıralanabilir. Ana çizgeler hazırlanırken internet çizge modeli kullanılmıştır. Bu modeldeki güç yasası (İng. Power law) adı verilen özellik gereği, çizgenin belli bölgelerinde yoğunluk artışları olabilir. Yani kullanılan çizgeler dengeli bir yoğunluğa sahip değildir. Ayrıca test çizgeleri tanımlanırken, çizge alanı sabit tutulmuş ve düğüm sayısı sürekli arttırılmıştır. Bu nedenle düğüm sayısı ile genel çizge yoğunluğu arasında doğru bir orantı bulunmaktadır. Bu orantı doğrultusunda/yoğunluk artışından dolayı, teorik olarak *Vc* çap sabitken ana çizge düğüm sayısına göre alt çizge boyutunda belli bir oranda artış gözlenmelidir.

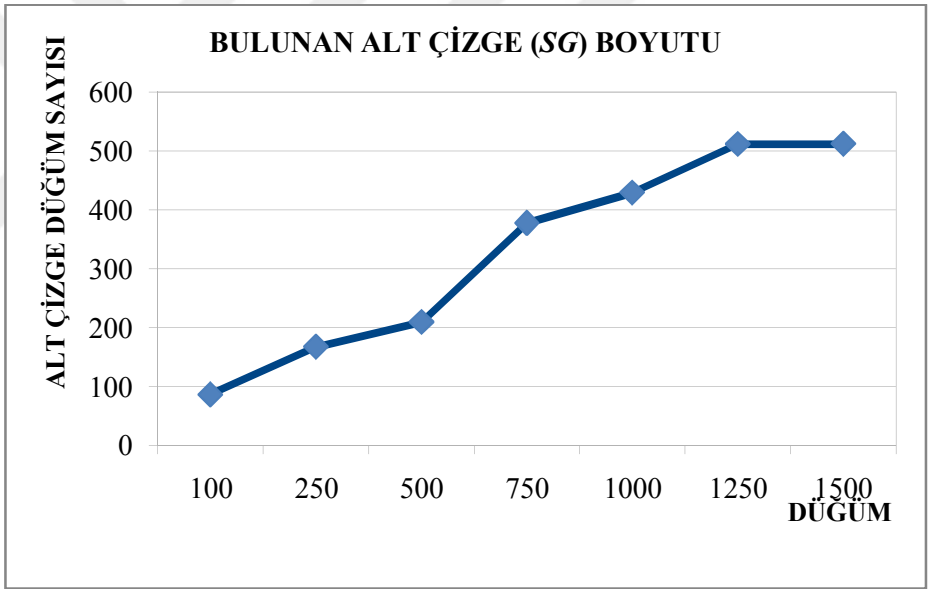
Test aşamasında *Vc* çapı 20 olarak alınmıştır. 100 ile 1500 arasında 7 farklı ana çizge kullanılmıştır. Her çizge için 5 farklı *Vc* set üzerinde deneme yapılmış ve elde edilen sonuçların ortalamaları alınarak aşağıdaki grafikler elde edilmiştir. Mesaj sayısı içine, her mesaj için en az bir kere gönderilmiş olan bilgilendirme (İng. acknowledgement) mesajları da dâhildir.



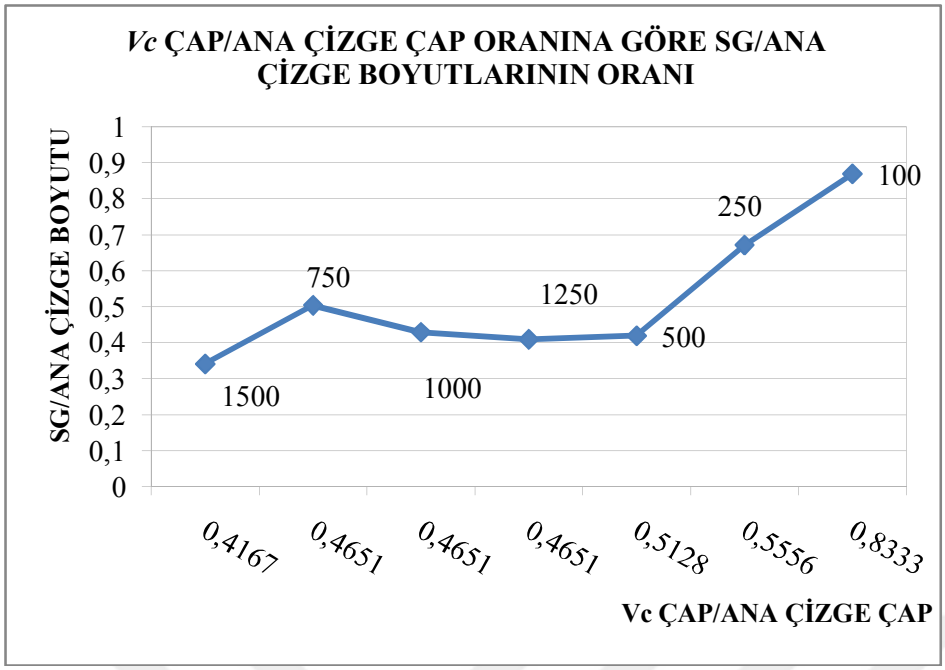
Şekil 4.4 *Vc* çapı sabit, çizge boyutu değişirken CCBC/*SG* belirleme algoritmasının çalışma süreleri.



Şekil 4.5 *Vc* çapı sabit, çizge boyutu değişirken CCBC/*SG* belirleme algoritmasının mesaj sayıları



Şekil 4.6 *Vc* çapı sabit, çizge boyutu değişirken belirlenen *SG* boyutu.



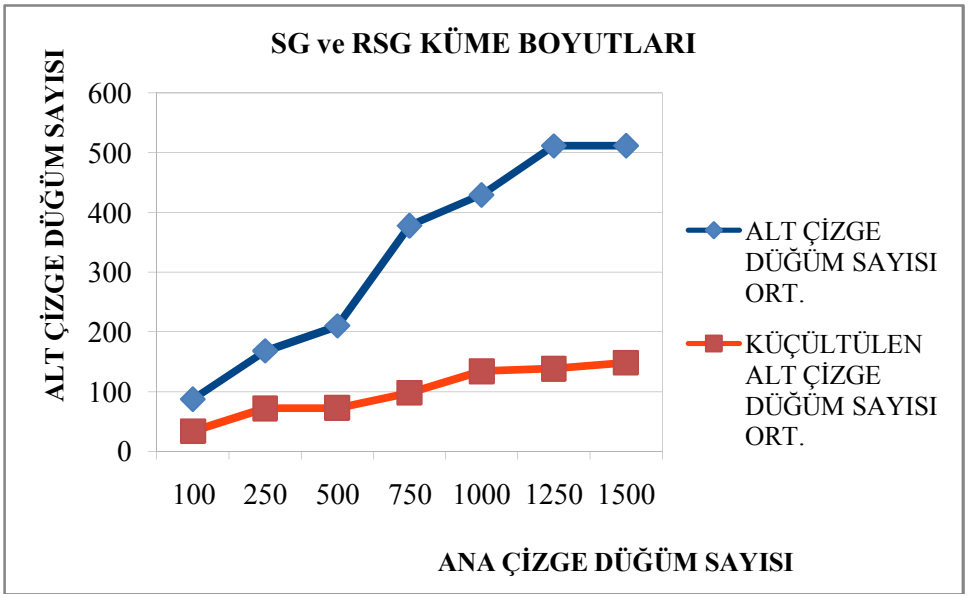
Şekil 4.7 *SG*/çizge boyutu ile *Vc* Çap/Çizge boyutu arasındaki ilişki.

Burada bir kez daha alt çizge boyutunun, *Vc* çaptan daha çok *Vc* çapın ana çizge çapıyla olan oranına ve *Vc* düğümlerin bulunduğu bölgenin yoğunluğuna bağlı olduğu gözlenmektedir. Çalışma zamanı düğüm sayısındaki artışa oranla daha az artarken, mesaj sayısının çok daha hızlı arttığı gözlenmektedir. Aslında çalışma süresinin, sabit *Vc* çap seçilmesinden ya da karmaşıklığının *O(d)* olmasından dolayı değişmemesi beklenmekteydi. Ama artan mesaj sayısı ile tamponlarda meydana gelen gecikmelerin bu durumda etkili olduğu düşünülmektedir.

Sonuç olarak alt çizge belirleme algoritması kullanımında, *Vc*_çap/ana_çizge_çap oranı sabit kaldığı sürece, *SG*_boyutu/ana_çizge_boyutu oranının da sabit kadığı görülmektedir. Yani *SG* boyutu *Vc*_ çap/ana_çizge_çap oranıyla yakından ilgilidir.

4.1.3 *SG* ile *RSG* kümeleri arasındaki ilişki

KDCBRA ve BCBRA algoritmalarının karmaşıklık değeri diğer algoritmalrdan çok daha yüksektir. Bu algoritmalarının verimini arttırmak amacıyla *RSG* kümesi tanımlanmıştır. *RSG* kümesinin boyutunu ve başarısını test etmek amacıyla testler yapılmıştır. Test aşamasında *Vc* çapı 20 olarak alınmıştır, 100 ile 1500 arasında 7 farklı ana çizge kullanılmıştır. Her çizge için 5 farklı *Vc* set üzerinde deneme yapılmış ve elde edilen *RSG* ve *SG* boyutlarının ortalamaları alınarak aşağıdaki grafikler elde edilmiştir.



Şekil 4.8 Vc çapı sabit, çizge boyutu değişirken *SG* ve *RSG* küme boyutları.

Bu testlerde *SG* boyutunun ana çizge boyutunun ortalama %52 si civarında, *RSG* boyutunun ise ortalama %18 civarında olduğu hesaplanmıştır. Grafikten de gözlenebileceği gibi ana çizge boyutu büyüdükçe, *SG* ve *RSG* boyutlarının ana çizge boyutuna olan oranları giderek düşmektedir. Bunu nedeni *Vc*_çap/Ana_çizge_çap oranın küçülmesidir. Sonuç olarak grafiklerdeki yataylaşmadan dolayı ana çizge boyutu büyüdükçe, *SG* ve *RSG* boyutlarındaki büyüme çok daha az olacaktır. Bu durum ana çizge boyutundaki artışın kaynak atama algoritmalarının performasları üzerindeki olumsuz etkisini azaltacaktır.

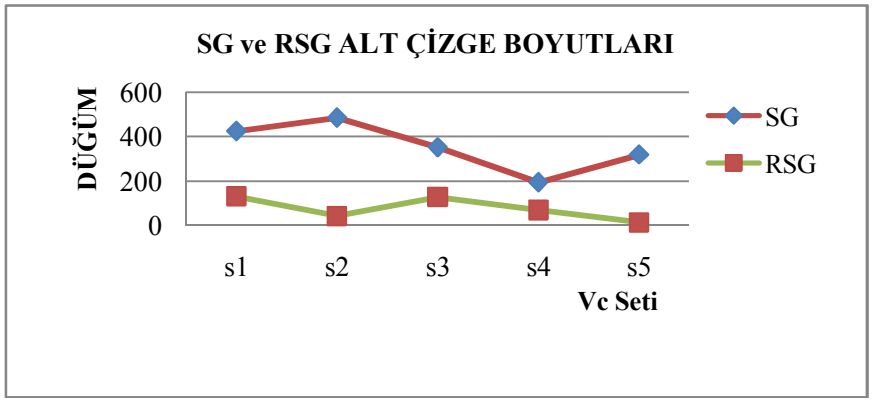
4.1.4 *RSG* Kümesinin doğruluk testi

Bu aşamada *RSG* kümesi içinden belirlenen adayların uygunluğunu test etmek amacıyla, bir benzetim çalışması yapılmış ve elde edilen veriler değerlendirilmiştir. Mesaj sayısı toplamına, her mesaj için en az bir kere gönderilmiş olan bilgilendirme mesajları da dâhildir. Benzetim verileri aşağıdaki adımlarla elde edilmiştir:

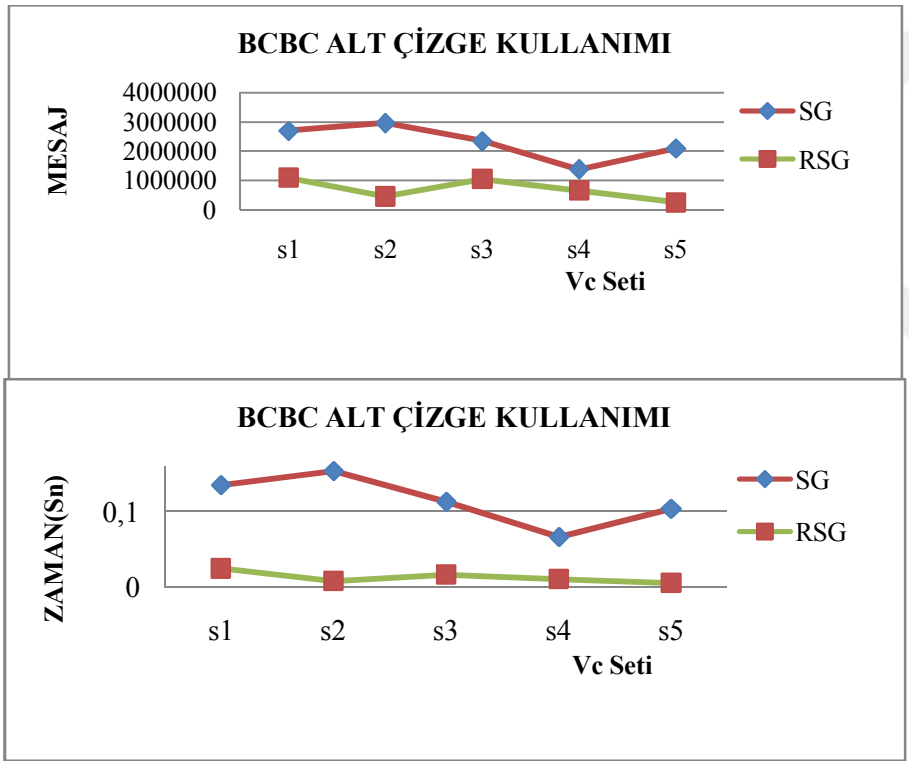
i. Ana çizge boyutu sabit (750 düğüm) iken seçilen 5 farklı *Vc* set (*Vc* çapı sabit 20 iken) üzerinde yakınlık merkezilik algoritması çalıştırılmış ve alt çizgeler belirlenmiştir (*SG*).

ii. Bu alt çizgelerden daha önce bahsedilen yaklaşımla indirgenmiş alt çizgeler (*RSG*) belirlenmiştir.

iii. KDCBC ve BCBC algoritmaları, her *Vc* sete ait hem *SG* hem de o *SG* den elde edilen *RSG* kümeleri üzerinde çalıştırılmıştır.



Şekil 4.9 750 dğümlü ana çizgeden elde edilne *SG* ve *RSG* kümeleri.



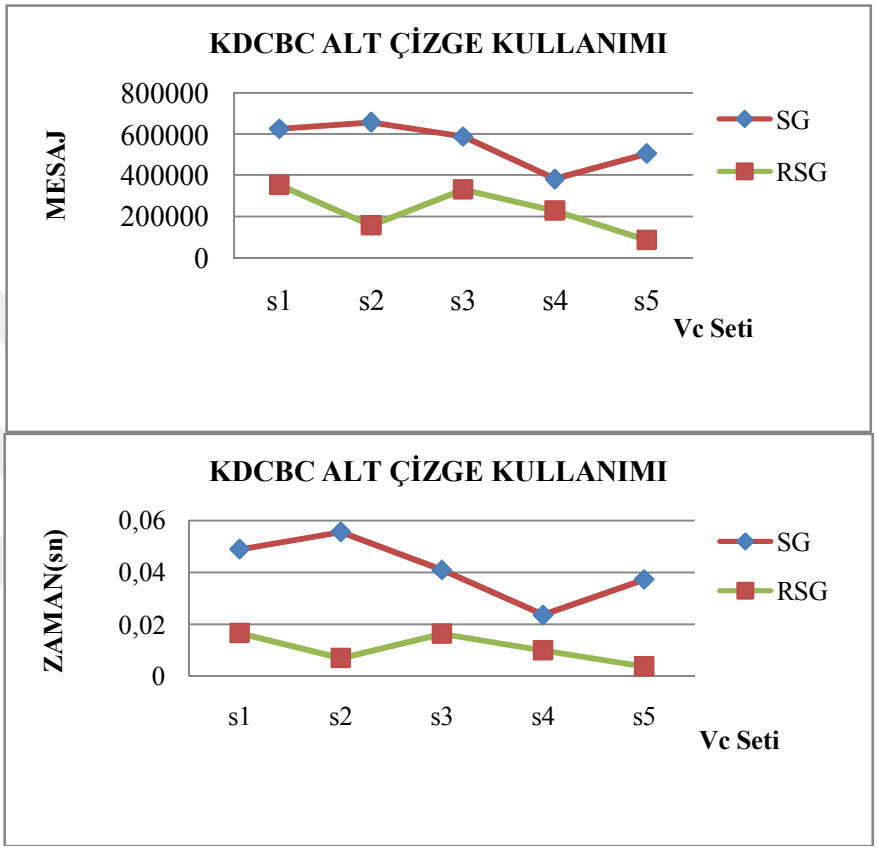
Şekil 4.10 BCBC algoritmasının *SG* ve *RSG* kullanarak aday belirleme sürecinde harcadıkları süre ve mesaj sayısı.

İndirgenmiş alt çizge ile bulunan adaylar, *SG* de bulunan adaylar ile kıyaslandığında,

- 2 sette ilk beş adayın sıralaması aynı,
- 1 sette ilk iki aday sıralaması aynı,
- 1 sette ilk adaylar aynı,
- 1 sette birinciler farklı bulunmuştur.

Bu aşamada birincileri farklı olan durum için her iki adayın da veri iletim testi yapılmıştır. Test sonucunda indirgenmiş alt çizge ile bulunan adayın veri iletim süresi, diğerinden daha iyi sonuç vermiştir.

- İndirgenmemiş alt çizge adayı 4: veri iletim süresi 18.05651474sn.
- İndirgenmiş alt çizge adayı 18: veri iletim süresi 12.02812195sn.



Şekil 4.11 KDBC algoritmasının SG ve RSG kullanarak aday belirleme sürecinde harcadıkları süre ve mesaj sayısı.

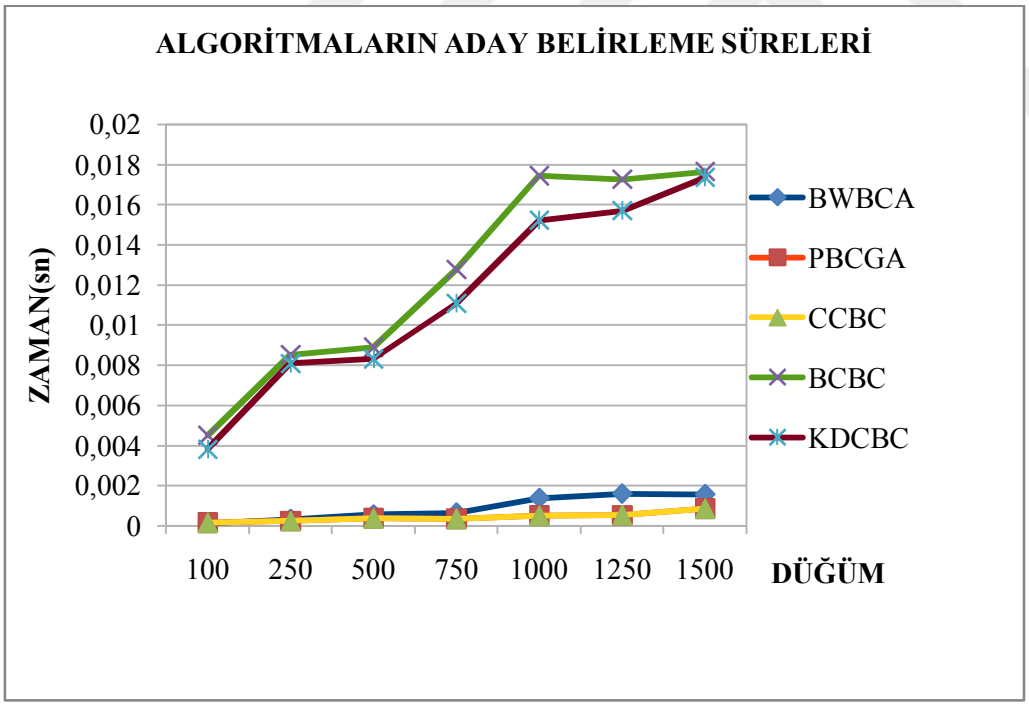
İndirgenmiş alt çizge ile bulunan adaylar, indirgenmemiş ile bulunan adaylar ile kıyaslandığında;

- 3 sette ilk 10 dan fazla adayın sıralaması aynı,
- 1 sette ilk 7 adayın sıralaması aynı,
- 1 sette ilk 5 adayın sıralaması aynı bulunmuştur.

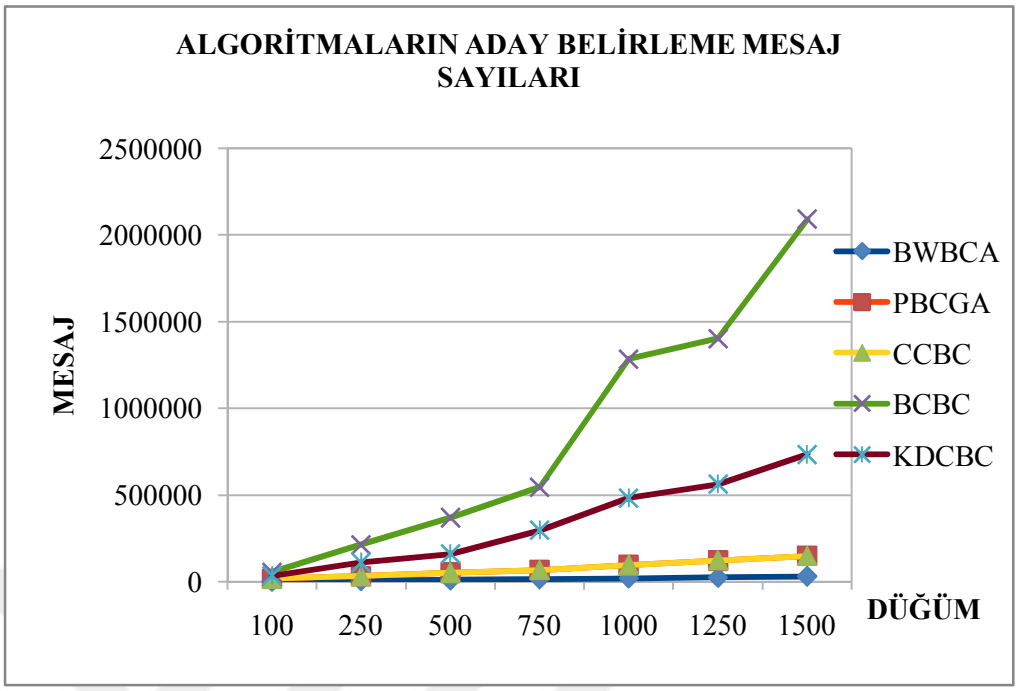
Bu sonuçlar, indirgenmiş alt çizge kullanımının en iyi adayı belirlemede ve işlem maliyetlerini düşürmede etkili olduğunu bize benzetim ortamında göstermiştir.

4.2 Aday Belirleme Algoritmalarının Aday Belirleme Benzetim Sonuçları

Bu aşamada algoritmaların aday belirleme maliyetlerini hesaplamak amacıyla testler yapılmıştır. Test verileri 100 ile 1500 düğüm arasında 7 ana çizgeden ve bu çizgelerden rastgele elde edilen *VC* kümelerinden oluşmaktadır. Her çizgeye ait 5 farklı *VC* kümesi belirlenmiştir. Her *VC* kümesi *20* adet *Vc* içermektedir ve her kümenin *Vc* çapı 20 birimdir. Önerilen (CCBC, KDCBC, BCBC) ve kıyaslama (PBCGA, BWBCA) amaçlı kodlanan kaynak atama algoritmaları, bu çizgelere ait her bir *Vc* seti için çalıştırılmış ve kendi aday listelerini oluşturmaları sağlanmıştır. Bu aşamada her bir algoritmaya ait çalışma süreleri ile iletilen mesaj sayıları ölçülerek aşağıdaki grafikler oluşturulmuştur. KDCBC ve BCBC algoritmaları aday belirleme işlemlerinde indirgenmiş alt çizge kullandıkları için bu algoritmaların zaman ve mesaj maliyetlerine CCBC algoritmasının zaman ve mesaj maliyetleri ilave edilmiştir. Ayrıca mesaj sayısı toplamına, her mesaj için en az bir kere gönderilmiş olan bilgilendirme mesajları da dâhiledilmiştir.



Şekil 4.12 Çizge boyutu değişirken algoritmaların aday belirleme süreleri.



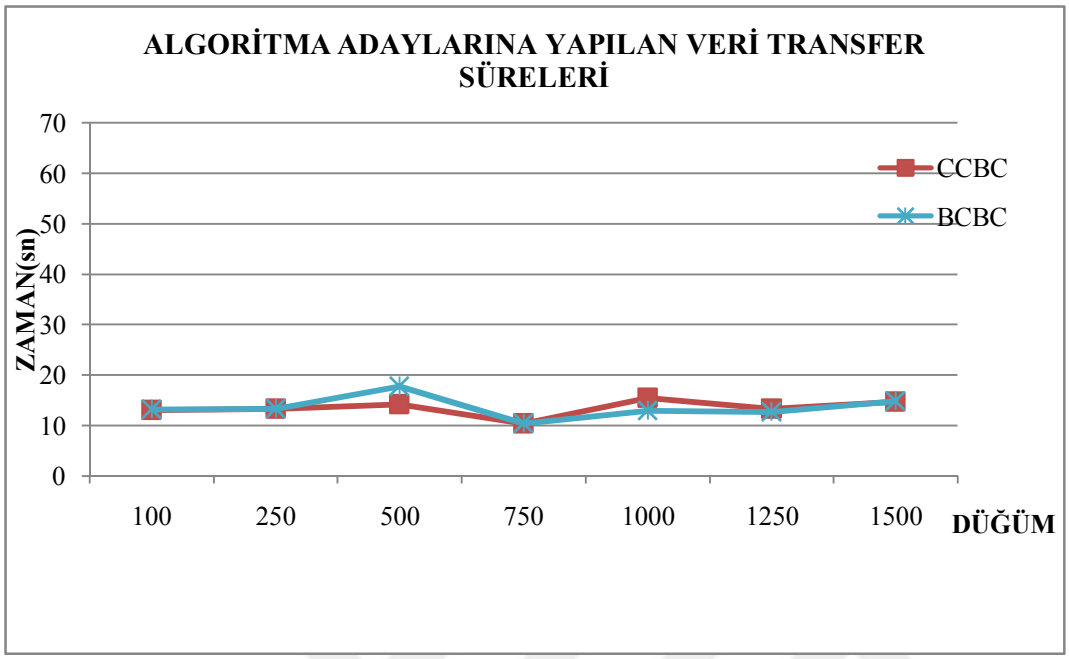
Şekil 4.13 Çizge boyutu değişirken algoritmaların aday belirlemede kullandıkları mesaj sayıları.

Bu aşmada teorik analizlere paralel benzetim sonuçları elde edilmiştir. *RSG* kümesi kullanılmasına rağmen, KDCBC ve BCBC algoritmalarının zaman ve mesaj maliyetlerinin diğerlerinden çok daha fazla olduğu görülmektedir. CCBC ve PBCGA algoritmaları, birbirlerine çok yakın alt yapıya sahip olamalarından dolayı neredeyse aynı mesaj ve zaman maliyetine sahiptir. Ama üretikleri aday listeleri oldukça farklıdır. BWBCA ise tüm çizge bazında çalışmasına rağmen algoritmasının basitliğinden dolayı en düşük maliyetli algoritmalardan biridir.

4.3 Algoritmaların Adaylarının Performans Test Sonuçları

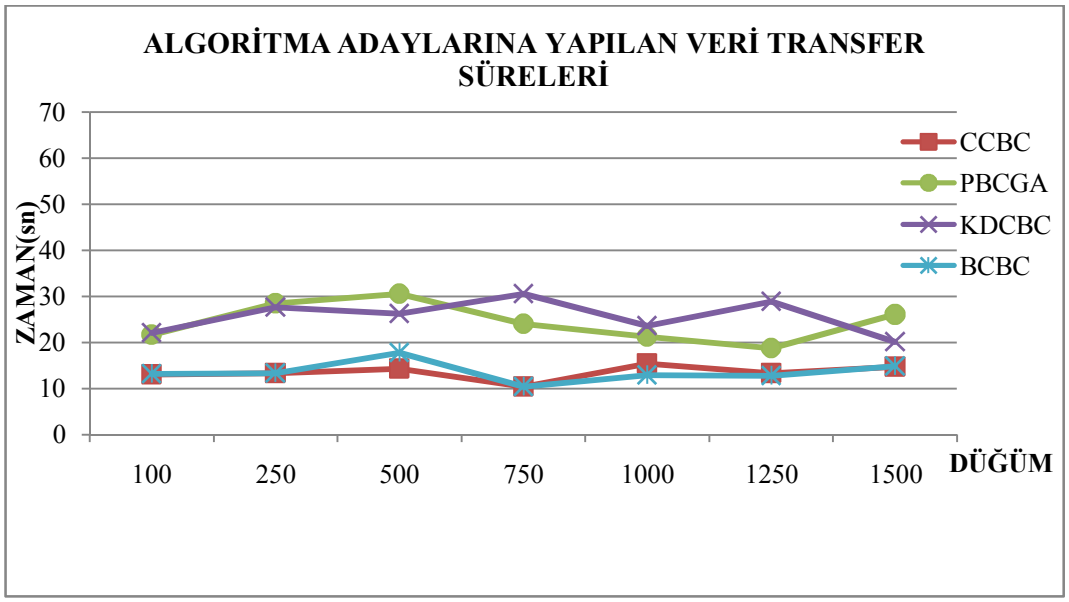
Bu aşamada CCBC, KDCBC, BCBC, PBCGA ve BWBCA algoritmalarınca belirlenen adayların, sorgu işleme anındaki performanslarını ölçmek amacıyla testler yapılmıştır. Daha önceki kabullenmelerden yola çıkarak sadece bir tane birleştirme operatörüne sahip olan sorgu birleştirme sistemi benzetimlenmiştir. Bu amaçla hazırlanan QJSA algoritması kullanılmıştır. Veri birleştirme operatörüne, her bir veri kaynağından 100MByte olmak üzere toplam 2GByte veri transferi yapılmış ve işlem süreleri ölçülmüştür. Algoritmaların, tüm çizgelere ait *VC* kümelerini kullanarak ürettiği adayların en iyileri veri birleştirme operatörü olarak

kullanılmıştır. Her çizgeye ait, *Vc* kümesi bazında ortalamalar alınarak grafikler oluşturulmuştur.



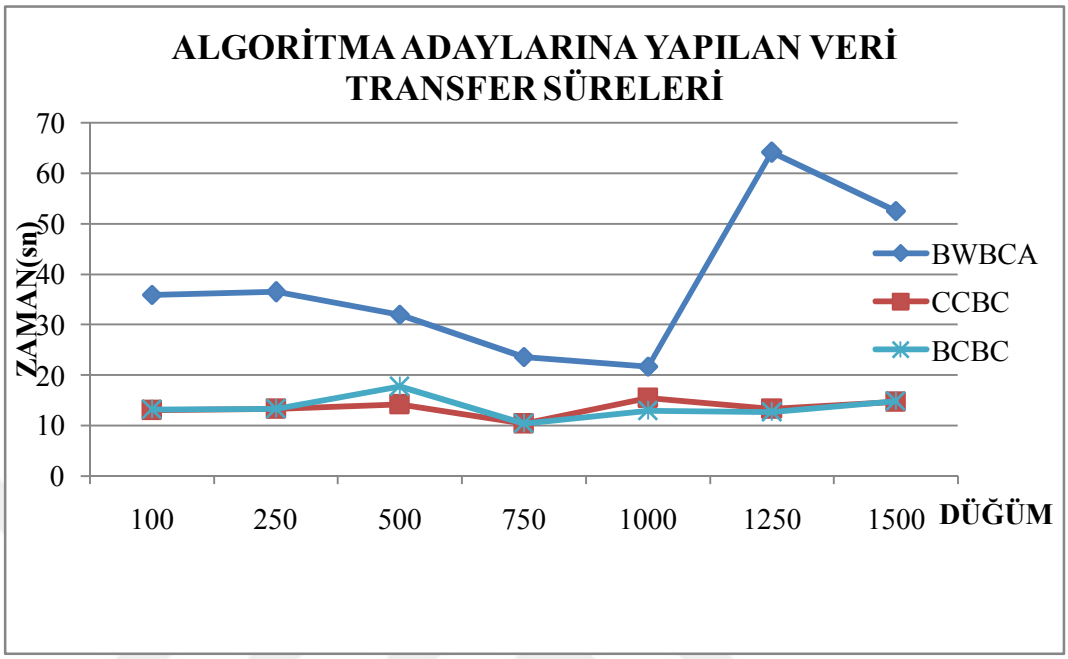
Şekil 4.14 Veri birleştirme benzetiminde, en iyi sonuç veren CCBC ve BCBC algoritmalarına ait adayların veri transfer süreleri.

CCBC ve BCBC algoritmalarına ait adaylar birbirlerine çok yakın sonuçlar üretmiş ve diğer algoritma adaylarından daha hızlı veri transferi sağlamışlardır.



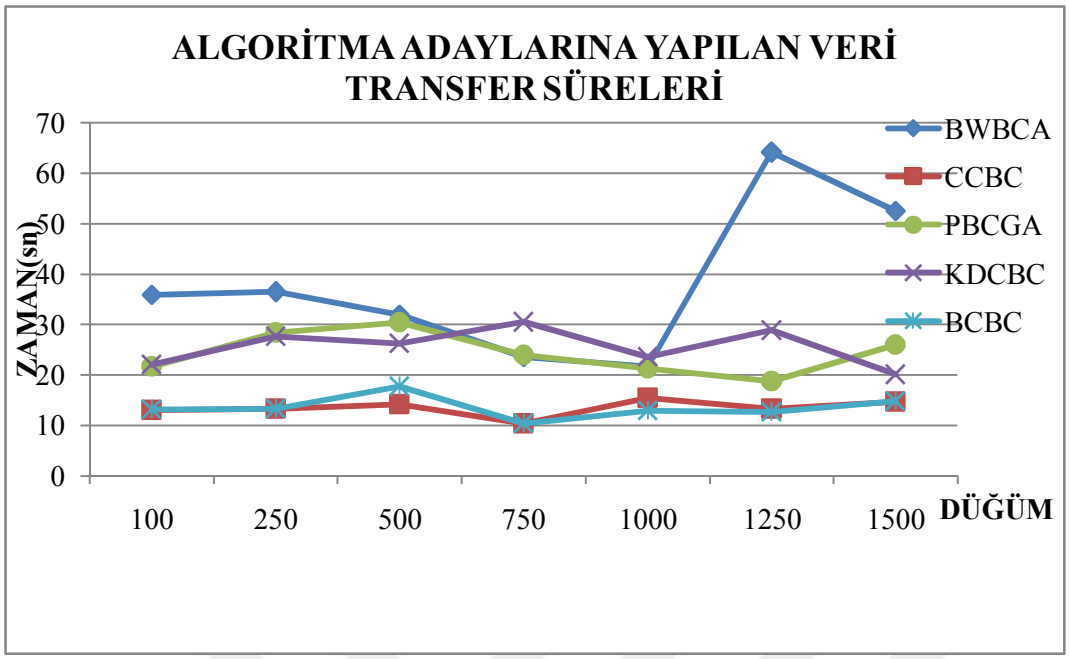
Şekil 4.15 Veri birleştirme benzetiminde, ikinci derece başarılı olan PBCGA ve KDCBC algoritmalaları ile en iyi sonuç veren algoritmalara ait adayların veri transfer süreleri.

KDCBGA ve PBCGA algoritmalarına ait adaylar, bir öncekilerden her zaman daha yavaş veri transferi yapmakla birlikte, birbirlerine yakın ama daha karasız sonuçlar üretmiştir.



Şekil 4.16 Veri birleştirme benzetiminde, en kötü sonuç veren BWBCA algoritması ile en iyi sonuç veren algoritmalara ait adayların veri transfer süreleri.

BWBCA algoritmasının adayları çoğunlukla diğer algoritmalara ait adaylardan daha yavaş veri transferi yapmakla birlikte, son derece karasız bir performans göstermiştir. Bunun en temel nedeni, bu algoritmanın adayını ana çizge genelinde belirlemesidir. Ana çizgenin her hangi bir yerindeki düğüm en yüksek kanal genişliği ortalamasına sahip olabilir ama veri kaynaklarının yakınında bulunan daha düşük kanal genişliği ortalamasına sahip bir başka düğüm daha iyi performans gösterebilmektedir. Bu grafik, adayların alt çizge içindeki düğümlerden seçilmesinin ne kadar doğru bir yaklaşım olduğunu göstermesi açısından oldukça önemlidir.

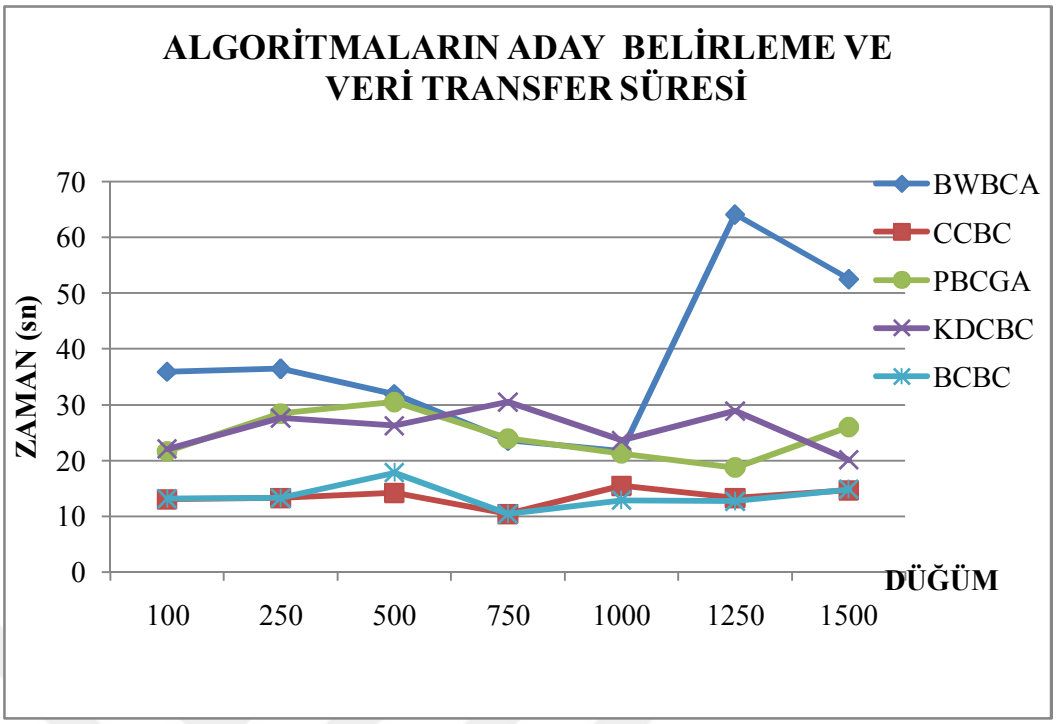


Şekil 4.17 Veri birleştirme benzetiminde, tüm algoritmalara ait adayların veri transfer süreleri.

Tüm algoritmalara ait eğrilere birlikte bakıldığında, bu çalışmada önerilen merkezilik tabanlı BCBC ve CCBC algoritmaların performansının diğerlerinden daha kararlı ve daha hızlı olduğu görülmektedir. Kıyaslama algoritması olarak kullandığımız BWBCA ve PGCGA algoritmalarının ise önerilen algoritmalara göre her zaman daha yavaş ve karasız olduğu görülmüştür. Sonuç olarak ilingesel kısıtları dikkate alan aradalılık ve yakınlık merkezilik tabanlı yaklaşımlar, veri birleştirme işlemlerinde diğer yaklaşımlardan daha iyi benzetim sonuçları vermiştir.

4.4 Baştan Sona Veri Birleştirme Süreci

Bu aşamada aday belirleme ve sorgu birleştireme işlemi için yapılan işlemler birleştirilerek sürecin top yekûn tamamlanma süresi ölçülmüştür. Bu işlem algoritmalara ait aday belirleme sürelerinin ilgili adayların veri transfer sürelerine ilave edilmesi ile gerçekleştirilmiştir.



Şekil 4.18 Algoritmaların baştan sona testi (aday belirleme ve veri transfer süresi birlikte).

Algoritmalara ait aday belirleme süreleri mili saniyeler mertebesindedir, bu süre toplam veri transfer süresinin binde biri civarındadır. Bu nedenle top yekûn sonuç eğrileri ile sadece veri transferini gösteren eğriler arasında belirgin bir fark oluşmamıştır. Bu durum toplam sorgu işleme sürecinde, veri transfer zamanın kaynak atama zamanında daha etkili olduğunu göstermiştir. Bir başka deyişle tetransfer edilecek veri miktarı arttıkça kaynak atama algoritmalarının maliyeti daha önemsiz hale gelecektir. Ayrıca birbirlerine çok yakın veri transfer performansı gösteren CCBC ve BCBC arasından en iyi olanı belirleyebilmek için aday belirleme aşamasındaki mesaj ve zaman maliyetlerine bakmak gerekmektedir (Bkz. Şekil 4.12 ve 4.13). Bu durumda, CCBC'nın en iyi sonuç veren algoritma olduğu görülmektedir.

5. SONUÇ

Bu tezde dağıtık sorgu işleme sistemlerinin kaynak atama aşamasında kullanılabilecek, merkezilik tabanlı algoritmalar üzerine bir çalışma yapılmıştır. Merkezilik tabanlı algoritmaların ve dolayısı ile düğümlere ait ilingesel özelliklerin sorgu işleme sisteminin performansı üzerine olan etikleri teorik ve pratik olarak incelenmiştir. Yapılan çalışma bir eniyileme işlemidir ve polinom zamanda çözümü olmadığı ispatlanmış bir probleme aittir. Bu nedenle sezgisel bir yaklaşımla en iyiye yakın çözümler önerilmiştir. Öncelikle bulunabilen dağıtık mimarideki merkezilik algoritmaları incelenerek detaylı bir analiz çalışması yapılmış ve karmaşık ağlar için uygunluğu incelenmiştir. Daha sonra dağıtık sorgu işleme sisteminin ihtiyaçları dikkate alınarak tasarlanan üç adet merkezilik tabanlı kaynak atama algoritması önerilmiş ve teorik analizleri yapılmıştır. Son olarak, hazırlanan benzetim ortamında, önerilen algoritmalar ile kodlanan kıyas algoritmaları çalıştırılarak performans analizleri yapılmıştır. Sonuç olarak bu tez çalışmasında elde edilen bulgular genel olarak aşağıda listeleniştir.

i. **Araştırma çalışması:** Bulanabilen dağıtık merkezilik tabanlı algoritmalar üzerine detaylı bir araştırma çalışması yapılmıştır. Çalışmalar daha önce belirlenen ölçütler (ölçeklenebilirlik, dinamiklik, güvenilirlik, zaman ve mesaj karmaşıklığı) dikkate alınarak karmaşık ağ analizine uygunlukları açısından değerlendirilmiştir. İncelenen çalışmaların zayıf ve güçlü yanları vurgulanmış, ayrıca temel özellikleri dikkate alınarak bir sınıflandırma çalışması yapılmıştır. Algoritmaların değerlendirmeleri, her sınıfın kendi içinde yapılmış ve sınıflara ait sonuçlar çizelgeler halinde sunulmuştur. Bu sayede daha sonra merkezilik üzerine çalışacak araştırmacılar için bir kaynak oluşturulmuştur. Bildiğimiz kadarı ile dağıtık merkezilik algoritmaları üzerine yapılan ilk tarama ve sınıflandırma çalışması yapılmıştır

Araştırmamız sonucunda, bu alanda var olan çalışma sayısının oldukça sınırlı olduğu gözlenmiştir. İncelenen çalışmalarda karmaşıklık düzeylerini düşürmek amacıyla pek çok sezgisel yaklaşım kullanılmakla birlikte, genellikle yerel veri kullanılmıştır. Ayrıca kullanılan yerel veri boyutu büyüdükçe ve genel çizge verisi ile arasındaki ilişki düzeyi arttıkça, elde edilen sonuçların güvenilirliğinin de arttığı görülmüştür. İncelenen çalışmaların çoğu ölçeklenebilir olmakla birlikte; genel olarak bilinirlik sınıfına ait çalışmalar en düşük, geçirgenlik sınıfına ait çalışmalar ise en büyük karmaşıklık düzeyine sahiptir.

Ayrıca bulunabilen çalışmalardan sadece birinin dinamiklik özelliğini içerdiği gözlenmiştir.

Kaynak atama işlemi üzerine yapılan araştırmada, aday listesi belirlenirken düğümlerin ilingesel özelliklerinden ziyade donanım ve yazılım özelliklerinin dikkate alındığı görülmüştür. Sadece bir çalışmada (Çokuslu et al. 2012a) yakınlık parametresinin kullanıldığı, ağ parametresi olarak ise sadece kanal genişliğinin (Gounaris et al. 2006b) dikkate alındığı gözlenmiştir. Bu durum bizi, ilingesel özelliklerin etkileri üzerine bir araştırma yapmaya yönlendirmiştir.

***ii.*Kaynak atama algoritmaları:** Araştırma çalışması sonucunda elde edilen bilgiler doğrultusunda, ilingesel etkileri kullanarak sonuç üreten üç adet merkezilik tabanlı dağıtık aday belirleme algoritması önerilmiştir. Önerilen algoritmaların, daha önce tanımlanan sınıfların iyi birer temsilcisi olmasına özen gösterilmiştir. Bu algoritmalar, sorgu işleme ve kaynak atama işleminin özellikleri dikkate alınarak tasarlanmıştır ve sırasıyla; Yakınlık Merkezilik Tabanlı Kaynak Atama (CCBC), *k*-uzaklık Merkezilik Tabanlı Kaynak Atama (KDCBC), ve Aradalılık Tabanlı Kaynak Atama (BCBC) algoritması olarak adlandırılmıştır.

Tasarlanan algoritmaların hepsinde sınırlı merkezilik adını verdiğimiz bir yaklaşım uygulanmıştır. Bu yaklaşım, merkezilik hesaplama ve sorgu işleme süreçlerinin özellikleri dikkate alınarak oluşturulmuştur ve bu çalışmanın önemli katkılarından birisidir. Böylece önerilen algoritmaların hem karmaşıklık düzeyleri önemli ölçüde düşürülmüş hem de doğru düğümlerin aday olarak seçilmesi sağlanmıştır.

Bu çalışmanın bir diğer katkısı ise alt çizge kullanımıdır. Önerilen algoritmaların, karmaşık ve büyük bir ağın tamamı üzerinde çalışmasını engellemek amacıyla *SG* ve *RSG* adını verdiğimiz alt çizgeler kullanılmıştır. *SG* ve *RSG* kümeleri CCBC algoritması tarafından belirlenmektedir ve daha maliyetli olan KDCBC ve BCBC algoritmalarının karmaşıklıklarını düşürmek için kullanılmaktadır. Bu sayede; seçilen veri kaynaklarının sayısı ve ağ üzerindeki yerleşimine bağlı olarak işlem maliyeti düşürülmüştür, veri trafiğinin ağın sadece ilgili bölümünde kalması sağlanmıştır, sistemin ölçeklenebilirliği arttırılmıştır ve en önemlisi, tüm çizge yerine sadece ilgili düğümler arasından (alt çizge içinden) seçim yaparak daha doğru adayların daha hızlı belirlenmesi sağlanmıştır.

Benzetim ortamında yapılan çalışmalarda, önerilen ve kıyas amaçlı kodlanan kaynak atama algoritmaları kullanılmıştır. Benzetim ortamı hazırlanırken, gerçek hayat şartlarına uygun olmasına özen gösterilmiş ve ilingesel etkileri daha net analiz etmemizi sağlayacak düzenlemeler yapılmıştır. Kıyas algoritmaları daha önce tanımlanan kabullenmeler doğrultusunda ve güncel çalışmalar dikkate alınarak kodlanmıştır. Bu sayede üç adet merkezilik yaklaşımı ile iki adet kıyas algoritmasının aynı ortamda ve aynı veriler üzerinde test edilmesi sağlanmıştır. Kaynak atama algoritmalarınca belirlenen en iyi adaylar, sorgu işleme sürecini benzeştirmek amacı ile tasarlanan Sorgu Birleştirme Benzetim (QJSA) algoritması ile değerlendirilmiştir. Elde edilen sonuçlar yorumlanıp grafikler halinde sunulmuştur.

Sonuç olarak sorgu işleme sistemindeki düğümlerin ilingesel özelliklerinin sorgu işleme performansı üzerindeki etkisini araştırmak üzere yapılan bu çalışmada, benzetim sonuçları ile teorik çalışmaların sonuçları birbirlerine ile paralel olduğu gözlenmiştir. Yakınlık ve aradalılık tabanlı CCBC ve BCBC algoritmalarının adayları, veri birleştirme aşamasındaki tüm denemelerde hem en iyi performansı göstermiş hem de her zaman diğerlerinden daha iyi sonuçlar üretmişlerdir. Bu durum yakınlık ve aradalılık merkezilik tabanlı yaklaşımların, kaynak atama sürecinde olumlu ve benzer oranda etkili olduklarını ve güvenilir olduklarını göstermiştir. Diğer algoritmalar ise bunlardan daha uzun sürede veri birleştirme işlemini tamamlamıştır. Kanal genişliği tabanlı BWBCA algoritmasının ise en uzun sürelerde sonuç üreten ve oldukça kararsız bir algoritma olduğu görülmüştür. Bu sonuçlara bakılarak ilingesel özelliklerin, sorgu işleme sisteminin veri birleştirme aşamasında dikkate alınması gereken bir kısıt olduğu sonucuna varılmıştır. Özellikle yakınlık merkezilik ölçütü, algoritma maliyetleri de dikkate alındığında, en iyi yaklaşım olarak ön plana çıkmıştır. Bu nedenle veri birleştirme operatörlerinin seçiminde kullanılan maliyet modeli içine diğer parametrelerle birlikte özellikle yakınlık merkezilik değerinin eklenmesi, doğru adayların seçiminde etkili olacak ve sorgu işleme sürecinde performans artışı sağlayacaktır.

Merkezilik tabanlı algoritmalar, kimi zaman başka isimler adı altında da olsa araştırmacılar tarafından uzun zamandır bilinmekte ve kullanılmaktadır. Bununla birlikte her geçen gün daha da büyüyen ve karmaşıklaşan ağların (sosyal/bilgisayar ağları), analizine olan ihtiyaç merkezilik yaklaşımını daha da ön plana çıkarmıştır. Bu nedenle gelecekte büyük ağların analizi üzerine, özellikle dağıtık merkezilik tabanlı algoritmalarının tasarlanması hedeflenmektedir.

KAYNAKLAR DİZİNİ

Barthélemy, M., 2004, Betweenness centrality in large complex networks, European Physical Journal B, 38, 163-168 pp.

Basagni, S., 1999, Distributed Clustering for Ad Hoc Networks, Proceedings of the 1999 International Symposium on Parallel Architectures, Algorithms and Networks, IEEE Computer Society: 310 p.

Batagelj, V. and M. Zaveršnik, 2011, Fast algorithms for determining (generalized) core groups in social networks, Advances in Data Analysis and Classification, 5, 2, 129-145 pp.

Bonacich, P., 1972, Factoring and weighting approaches to status scores and clique identification, Journal of Mathematical Sociology, 2, 1, 113-120 pp.

Borbash, S. A., A. Ephremides and M. J. McGlynn, 2007, An asynchronous neighbor discovery algorithm for wireless sensor networks, Ad Hoc Netw., 5, 7, 998-1016 pp.

Borgatti, S., 2005, Centrality and network flow, Social Networks, 27, 1, 55-71 pp.

Borgatti, S. P. and M. G. Everett, 2006, A Graph-theoretic perspective on centrality, Social Networks, 28, 4, 466-484 pp.

Bose, S. K., S. Krishnamoorthy and N. D. Ranade, 2013, Allocating resources for parallel execution of query plans, Google Patents.

Bradler, D., L. Krumov, M. Mühlhäuser and J. Kangasharju, 2011, BridgeFinder: Finding communication bottlenecks in distributed environments, ICOIN, C. Kim and Y. Shin, IEEE: 1-6 pp.

Brandes, U., 2001, A faster algorithm for betweenness centrality, Journal of Mathematical Sociology, 25, 163.

Can Umut, I., U. Cemil Aybars, D. Orhan and K. Vedat, 2016, On Vertex Cover Problems in Distributed Systems, Advanced Methods for Complex Network Analysis, M. Natarajan. Hershey, PA, USA, IGI Global: 1-29 pp.

Chihping, W., 1990, The complexity of processing tree queries in distributed databases,, in Parallel and Distributed Processing Proceedings of the Second IEEE Symposium on, 604-611 pp.

ChongGun, K. and W. Mary, 2013, Leader election on tree-based centrality in ad hoc networks, 52, 661-670 pp.

Çokuslu, D., 2012c, Resource Discovery and Allocation for Query Processing in Grid Systems, Phd.

Çokuslu, D., A. Hameurlain, K. Erciyes and F. Morvan, 2012b, Resource allocation algorithm for a relational join operator in grid systems, Proceedings of the 16th International Database Engineering & Applications Sysmposium, Prague, Czech Republic, ACM: 139-145 pp.

Çokuslu, D., A. Hameurlain and E. Kayhan, 2012a, Resource allocation for query processing in grid systems: a survey, Comput. Syst. Sci. Eng., 27, 4.

Dedzoe, W. K., P. Lamarre, R. Akbarinia and P. Valduriez, 2010, ASAP Top-k Query Processing in Unstructured P2P Systems, Peer-to-Peer Computing, IEEE: 1-10 pp.

KAYNAKLAR DİZİNİ (devam)

Eppstein, D. and J. Wang, 2001, Fast approximation of centrality, Proceedings of the twelfth annual ACM-SIAM symposium on Discrete algorithms, Washington, D.C., USA, Society for Industrial and Applied Mathematics: 228-229 pp.

Erciyes, K., 2013, Distributed Graph Algorithms for Computer Networks, Springer Publishing Company, Incorporated. 350 p

Erciyes, K., 2014, Complex Networks: An Algorithmic Perspective, CRC Press. 320 p

Freeman, L., 1979, Centrality in social networks: Conceptual clarification, Social Networks, 1, 3, 215-239 pp.

Freeman, L. C., 1977, A Set of Measures of Centrality Based on Betweenness, Sociometry, 40, 1, 35-41 pp.

Gounaris, A., R. Sakellariou, N. W. Paton and A. A. Fernandes, 2006a, A novel approach to resource scheduling for parallel query processing on computational grids, Distrib. Parallel Databases, 19, 2-3, 87-106 pp.

Gounaris, A., R. Sakellariou, N. W. Paton and A. A. A. Fernandes, 2004, Resource Scheduling for Parallel Query Processing on Computational Grids, Proceedings of the 5th IEEE/ACM International Workshop on Grid Computing, IEEE Computer Society: 396-401 pp.

Gounaris, A., J. Smith, N. W. Paton, R. Sakellariou, A. A. Fernandes and P. Watson, 2009, Adaptive workload allocation in query processing in autonomous heterogeneous environments, Distrib. Parallel Databases, 25, 3, 125-164 pp.

Gounaris, A., J. Smith, N. W. Paton, R. Sakellariou, A. A. A. Fernandes and P. Watson, 2006b, Adapting to Changing Resource Performance in Grid Query Processing, Data Management in Grids: First VLDB Workshop, DMG 2005, Trondheim, Norway, September 2-3, 2005, Revised Selected Papers, J.-M. Pierson. Berlin, Heidelberg, Springer Berlin Heidelberg: 30-44 pp.

Green, O., R. McColl and D. A. Bader, 2012, A Fast Algorithm for Streaming Betweenness Centrality, Proceedings of the 2012 ASE/IEEE International Conference on Social Computing and 2012 ASE/IEEE International Conference on Privacy, Security, Risk and Trust, IEEE Computer Society: 11-20 pp.

Guidi, B., M. Conti, A. Passarella and L. Ricci, 2014, Distributed protocols for Ego Betweenness Centrality computation in DOSNs: 539-544 pp.

Hameurlain, A., D. Cokuslu and K. Erciyes, 2010, Resource discovery in grid systems\&\#58; a survey, Int. J. Metadata Semant. Ontologies, 5, 3, 251-263 pp.

Hastings, W. K., 1970, Monte Carlo sampling methods using Markov chains and their applications, Biometrika, 57, 1, 97-109 pp.

Holme, P., 2003, Congestion and centrality in traffic flow on complex networks, Advances in Complex Systems (ACS), 06, 02, 163-176 pp.

KAYNAKLAR DİZİNİ (devam)

Hua, Q. S., H. Fan, M. Ai, L. Qian, Y. Li, X. Shi and H. Jin, 2016, Nearly Optimal Distributed Algorithm for Computing Betweenness Centrality, 2016 IEEE 36th International Conference on Distributed Computing Systems (ICDCS).

Hwang, W., Y.-r. Cho, A. Zhang and M. Ramanathan, 2006, Bridging Centrality: Identifying Bridging Nodes in Scale-free Networks.

Ilyas, M., Z. Shafiq, A. Liu and H. Radha, 2011, A distributed and privacy preserving algorithm for identifying information hubs in social networks, Proceedings of IEEE INFOCOM 2011, Shanghai, China, IEEE.

Ilyas, M. U. and H. Radha, 2010, A KLT-inspired node centrality for identifying influential neighborhoods in graphs, CISS, IEEE: 1-7 pp.

Jain, A. K., M. N. Murty and P. J. Flynn, 1999, Data clustering: a review, ACM Comput. Surv., 31, 3, 264-323 pp.

K Sheshadri, R. a. A., Mustafa Y. and Sundaresan, Karthikeyan and Rangarajan, Sampath and Koutsonikolas, Dimitrios, 2016, AmorFi: Amorphous WiFi Networks for High-density Deployments. Proceedings of the 12th International on Conference on Emerging Networking EXperiments and Technologies.

Kavalci, V., A. Ural and O. Dagdeviren, 2014, Distributed Vertex Cover Algorithms For Wireless Sensor Networks, CoRR, abs/1402.2140, 95-110 pp.

Kempe, D. and F. McSherry, 2008, A decentralized algorithm for spectral analysis, Journal of Computer and System Sciences, 74, 1, 70-83 pp.

Kermarrec, A.-M., E. L. Merrer, B. Sericola, G. Tr\, \#233 and dan, 2011, Second order centrality: Distributed assessment of nodes criticity in complex networks, Comput. Commun., 34, 5, 619-628 pp.

Kossmann, D., 2000, The state of the art in distributed query processing, ACM Comput. Surv., 32, 4, 422-469 pp.

Kotowski, N., A. A. B. Lima, E. Pacitti, P. Valduriez and M. Mattoso, 2008, Parallel query processing for OLAP in grids, Concurr. Comput. : Pract. Exper., 20, 17, 2039-2048 pp.

Lehmann, K. A. and M. Kaufmann, 2003, Decentralized Algorithms for Evaluating Centrality in Complex Networks, WSI.

Liu, S. and H. A. Karimi, 2008, Grid query optimizer to improve query processing in grids, Future Gener. Comput. Syst., 24, 5, 342-353 pp.

Lulli, A., L. Ricci, E. Carlini and P. Dazzi, 2015, Distributed Current Flow Betweenness Centrality, 2015 IEEE 9th International Conference on Self-Adaptive and Self-Organizing Systems, 71-80 pp.

Marsden, P. V., 2002, Egocentric and sociocentric measures of network centrality, Social Networks, 24, 4, 407-422 pp.

Medina, A., A. Lakhina, I. Matta and J. Byers, 2001, BRITE: An Approach to Universal Topology Generation, Proceedings of the Ninth International Symposium in Modeling, Analysis and Simulation of Computer and Telecommunication Systems, IEEE Computer Society: 346 p.

Merrer, E. L. and G. Trédan, 2009, Centralities: capturing the fuzzy notion of importance in social graphs, SNS, T. Stein and M. Cha, ACM: 33-38 pp.

KAYNAKLAR DİZİNİ (devam)

Montresor, A., F. D. Pellegrini and D. Miorandi, 2013, Distributed k-Core Decomposition, IEEE Trans. Parallel Distrib. Syst., 24, 2, 288-300 pp.

Nanda, S. and D. Kotz, 2008, Localized Bridging Centrality for Distributed Network Analysis, ICCCN'08, 62-67 pp.

Newman, M. E. J., 2000, Who is the Best Connected Scientist? A Study of Scientific Coauthorship Networks, Santa Fe Institute.

Newman, M. E. J., 2005, A measure of betweenness centrality based on random walks, Social Networks, 27, 1, 39-54 pp.

Newman, M. E. J. and M. Girvan, 2004, Finding and evaluating community structure in networks, Phys. Rev. E, 69, 2, 026113 p.

Nomikos George, P. P., Karaliopoulos Merkourios, Stavrakakis Ioannis 2013, The Multiple Instances of Node Centrality and their Implications on the Vulnerability of ISP Networks, CoRR, abs/1312.4707.

Noori, A., 2011, On the Relation between Centrality Measures and Consensus Algorithms High Performance Computing and Simulation (HPCS), 2011 International Conference on, abs/1208.1740, 225 - 232 pp.

The ns-3 network simulator. , 2008, from https://www.nsnam.org. (Access year: 2018)

Oliveira, E. M. R., H. S. Ramos and A. A. F. Loureiro, 2010, Centrality-based routing for Wireless Sensor Networks, Wireless Days, IEEE: 1-5 pp.

Ozsu, M. T., 2007, Principles of Distributed Database Systems, Prentice Hall Press. 704 p

P. Fekete, S., M. Kaufmann, A. Kröller and K. Zweig, 2005, A New Approach for Boundary Recognition in Geometric Sensor Networks, CoRR, abs/cs/0508006, 4 p.

Phillip Bonacich, P. L., 2001, Eigenvector-like measures of centrality for asymmetric relations, Social Networks, 23 (2001) 191-201 pp.

Ramos, H. S., A. C. Frery, A. Boukerche, E. M. R. Oliveira and A. A. F. Loureiro, 2014, Topology-Related Metrics and Applications for the Design and Operation of Wireless Sensor Networks, ACM Trans. Sen. Netw., 10, 3, 1-35 pp.

Sabidussi, G., 1966, The centrality index of a graph, Psychometrika, 31, 4, 581-603 pp.

Silva, V. F. V. D., M. L. Dutra, F. Porto, B. Schulze, A. C. Barbosa and J. C. d. Oliveira, 2006, An adaptive parallel query processing middleware for the Grid: Research Articles, Concurr. Comput. : Pract. Exper., 18, 6, 621-634 pp.

Sottile, P. F. a. F. D. R. a. C., 2016, A Predictive Cross-Layered Interference Management in a Multichannel MAC with Reactive Routing in VANET, IEEE Transactions on Mobile Computing, 15-8, 1850-1862 pp.

Stonebraker, M., P. M. Aoki, W. Litwin, A. Pfeffer, A. Sah, J. Sidell, C. Staelin and A. Yu, 1996, Mariposa: A Wide-Area Distributed Database System, VLDB J., 5, 1, 48-63 pp.

Tasoulis, D. K. and M. N. Vrahatis, 2005, Unsupervised clustering on dynamic databases, Pattern Recogn. Lett., 26, 13, 2116-2127 pp.

KAYNAKLAR DİZİNİ (devam)

Travieso, G., C. A. Ruggiero, O. M. Bruno and L. da F. Costa, 2013, Predicting efficiency in master–slave grid computing systems, Journal of Complex Networks, 1, 1, 63-71 pp.

Vasudevan, S., J. Kurose and D. Towsley, 2004, Design and Analysis of a Leader Election Algorithm for Mobile Ad Hoc Networks, Proceedings of the 12th IEEE International Conference on Network Protocols, IEEE Computer Society: 350-360 pp.

Wang, X. a. C., Guanrong, 2003, Complex networks: small-world, scale-free, and beyond, IEEE Circuits Syst Mag, v3 i1, 6-20 pp.

Wehmuth, K., 2012a, Distributed Assessment of Network Centralities in Complex Social Networks.

Wehmuth, K., A. G. Antonio Tadeu and Z. Artur, 2011a, DANCE: A Framework for the Distributed Assessment of Network Centralities, CORD Conference Proceedings.

Wehmuth, K. and A. Ziviani, 2011b, Distributed Assessment of Network Centrality, CoRR, abs/1108.1067.

Wehmuth, K. and A. Ziviani, 2012b, Distributed assessment of the closeness centrality ranking in complex networks, Proceedings of the Fourth Annual Workshop on Simplifying Complex Networks for Practitioners, Lyon, France, ACM: 43-48 pp.

Wei, W., X.-L. Yang, P. Shen and B. Zhou, 2012, Holes Detection in Anisotropic Sensornets: Topological Methods, IJDSN, 2012.

Xing, B., M. Deshpande, S. Mehrotra and N. Venkatasubramanian, 2010, Gateway designation for timely communications in instant mesh networks, PerCom Workshops, IEEE: 564-569 pp.

Zhuge, H. and J. Zhang, 2009, Topological Centrality and Its Applications, CoRR, abs/0902.1911.

ÖZGEÇMİŞ

Vedat KAVALCI
Adres: Ankara Cd. No:229, D:25, Bornova, İzmir.
Telefon: 0 542 2323091
Email: borbim@gmail.com
Milliyet: TC.
Doğum Yeri ve Tarihi: 1966.03.24, TÜRKİYE-ESKİŞEHİR

Eğitim Durumu:

Doktora: 2012-, Uluslararası Bilgisayar Enstitüsü, Ege Üniversitesi (Not Ort.: 3.49)

Yüksek Lisans: 1987 – 1991, Bilgisayar Mühendisliği, Ege Üniversitesi, İzmir/TÜRKİYE.

Lisans: 1983 – 1987, Bilgisayar Mühendisliği, Ege Üniversitesi, İzmir/TÜRKİYE.

Lise: 1980 – 1983, Sivrihisar Lisesi, Eskişehir/TÜRKİYE.

Yabancı Dil:

İngilizce: orta.

Yayınlar:

Tunali, T., Erciyes, K., Kavalci, V., 1989, A Local Network for A Multiprocessor Controller, Proc. of the Fourth International Symposium on Computer and Information Sciences, Izmir, Turkey, Nova Science Publishers Inc., Vol.1, Commack New York, 659-668 pp.

Kavalci, V., A. Ural and O. Dagdeviren, 2014, Distributed Vertex Cover Algorithms For Wireless Sensor Networks, CoRR, abs/1402.2140, 95-110 pp.

Can Umut, I., U. Cemil Aybars, D. Orhan and K. Vedat, 2016, On Vertex Cover Problems in Distributed Systems, Advanced Methods for Complex Network Analysis, M. Natarajan. Hershey, PA, USA, IGI Global: 1-29 pp.

Kavalci, V., Çokuslu D., 2018, Distributed centrality algorithms: A survey, Journal of Parallel and Distributed Computing., ELSEVIER (27/Apr/2018 , Decision Pending)..