**T.C.**
**PAMUKKALE ÜNİVERSİTESİ**
**FEN BİLİMLERİ ENSTİTÜSÜ**
**BİLGİSAYAR MÜHENDİSLİĞİ ANABİLİM DALI**

# TELSİZ DUYARGA AĞLARINDA AĞIRLIKLI BAĞLI BASKIN KÜME ALGORİTMALARI

**YÜKSEK LİSANS TEZİ**

**MUSTAFA TOSUN**

**DENİZLİ, TEMMUZ - 2018**

**T.C.**
**PAMUKKALE ÜNİVERSİTESİ**
**FEN BİLİMLERİ ENSTİTÜSÜ**
**BİLGİSAYAR MÜHENDİSLİĞİ ANABİLİM DALI**

# TELSİZ DUYARGA AĞLARINDA AĞIRLIKLI BAĞLI BASKIN KÜME ALGORİTMALARI

**YÜKSEK LİSANS TEZİ**

**MUSTAFA TOSUN**

**DENİZLİ, TEMMUZ - 2018**

## KABUL VE ONAY SAYFASI

**Mustafa TOSUN** tarafından hazırlanan "**TELSİZ DUYARGA AĞLARINDA AĞIRLIKLI BAĞLI BASKIN KÜME ALGORİTMALARI**" adlı tez çalışmasının savunma sınavı 27.07.2018 tarihinde yapılmış olup aşağıda verilen jüri tarafından oy birliği / oy çokluğu ile Pamukkale Üniversitesi Fen Bilimleri Enstitüsü Bilgisayar Mühendisliği Anabilim Dalı Yüksek Lisans Tezi olarak kabul edilmiştir.

| Jüri Üyeleri | İmza |
|---|---|
| Danışman<br>Dr. Öğr. Ü. Elif Haytaoğlu<br>Pamukkale Üniversitesi | ..... |
| Üye<br>Prof. Dr. Sezai TOKAT<br>Pamukkale Üniversitesi | ..... |
| Üye<br>Dr. Öğr. Ü. Moharram CHALLENGER<br>Ege Üniversitesi | ..... |

Pamukkale Üniversitesi Fen Bilimleri Enstitüsü Yönetim Kurulu'nun 08.08.2018 tarih ve ...32/16.. sayılı kararıyla onaylanmıştır.

.....

Prof. Dr. Uğur YÜCEL

Fen Bilimleri Enstitüsü Müdürü

**Bu tez çalışması Pamukkale Üniversitesi Bilimsel Araştırma Projeleri Koordinasyon Birimi tarafından 2018FEBE013 nolu proje ile desteklenmiştir.**

**Bu tezin tasarımı, hazırlanması, yürütülmesi, araştırmalarının yapılması ve bulgularının analizlerinde bilimsel etiğe ve akademik kurallara özenle riayet edildiğini; bu çalışmanın doğrudan birincil ürünü olmayan bulguların, verilerin ve materyallerin bilimsel etiğe uygun olarak kaynak gösterildiğini ve alıntı yapılan çalışmalara atfedildiğine beyan ederim.**

**Mustafa TOSUN**

# ÖZET


**TELSİZ DUYARGA AĞLARINDA AĞIRLIKLI BAĞLI BASKIN KÜME ALGORİTMALARI**

**YÜKSEK LİSANS TEZİ**

**MUSTAFA TOSUN**

**PAMUKKALE ÜNİVERSİTESİ FEN BİLİMLERİ ENSTİTÜSÜ**

**BİLGİSAYAR MÜHENDİSLİĞİ ANABİLİM DALI**

**(TEZ DANIŞMANI:DR. ÖĞR. Ü. ELİF HAYTAOĞLU)**

**DENİZLİ, TEMMUZ - 2018**

Telsiz duyarga ağları askeri, gıda, tarım vb. alanlarda sıkça kullanılmaktadır. Telsiz duyarga ağlarında iletişim için kullanılan enerji miktarını azaltmak ve ağın yaşam süresini artırmak için iletişim omurgası olarak bağlı baskın kümeler kullanılmaktadır. Kullanılan küme ne kadar küçük olursa o kadar az düğümün enerjisi azalmaktadır. Dolayısıyla bağlı baskın kümeler (CDS) oluşturulurken, bu kümenin küçük olması istenmektedir. Fakat bağlı baskın kümelerin küçük boyutlarda olması yeterli olmayabilmektedir. Bağlı baskın küme içerisinde enerjisi düşük olan düğümler seçilirse düğümün enerjisi daha çabuk biterek omurganın çalışmasını durdurabilmektedir. Bu yüzden bağlı baskın kümeye konulacak düğümler seçilirken kalan enerji seviyelerinin de dikkate alınması gerekmektedir. Bağlı baskın kümeyi oluşturan düğümlerin ağırlıklarının toplamları minimal olan bağlı baskın kümelere minimal ağırlıklı bağlı baskı küme (MWCDS) denilmektedir. Bu çalışmada iki yeni MWCDS algoritması önerilmiştir. Algoritmalardan ilki düğümlerin ağırlıklarının yanında düğümler arasında bulunan en kısa yolları da dikkate alarak CDS'i oluşturan minimal ağırlık ve yönlendirme maliyetli bağlı baskın küme (MWOC-CDS) algoritmasıdır. İkinci algoritma ise komşularına göre daha farklı düğümlere bağlanabilen kritik düğümleri dikkate alarak bağlı baskın kümeyi oluşturan kritik düğüm tabanlı minimal ağırlıklı bağlı baskın küme (CN-MWCDS) algoritmasıdır.



**ANAHTAR KELİMELER:Bağlı Baskın Küme, Telsiz Duyarga Ağları, Minimum Ağırlıklı Bağlı Baskın Küme, Minimal Ağırlıklı Bağlı Baskın Küme, Minimal Yönlendirme Maliyetli Bağlı Baskın Küme**

# ABSTRACT

**WEIGHTED CONNECTED DOMINATING SET ALGORITHMS IN WIRELESS SENSOR NETWORKS**
**MSC THESIS**
**MUSTAFA TOSUN**
**PAMUKKALE UNIVERSITY INSTITUTE OF SCIENCE**
**COMPUTER ENGINEERING**
**(SUPERVISOR:ASST. PROF. DR. ELİF HAYTAOĞLU)**

**DENİZLİ, JULY 2018**

Wireless sensor networks are often used in military, food, agriculture areas etc. To decrease energy amount consumed in the communication and increase lifetime of the network, connected dominating sets are used as a communication backbone in wireless sensor networks. Due to the total energy consumption of CDSs with few nodes is much less than CDSs with high cardinality the connected dominating sets are desired to have few nodes. However, decreasing the size of CDSs may not be enough. If nodes with low energy are selected in CDS, nodes stop working before other nodes by running out of energy. This situation causes the backbone to fail. Thus, while nodes are being choosed, their energy level should be considered. A CDS whose nodes' total weight is minimal, is named as the minimal weighted connected dominating set (MWCDS). In this work, two new MWCDS algorithms are proposed. The first one is the minimal weighted and routing cost connected dominating set (MWOC-CDS) algorithm which construct CDS by considering the shortest paths between two nodes as well as nodes' weights. The second one is critical node based the minimal weighted connected dominating set (CN-MWCDS) algorithm which construct CDS by considering critical nodes that can connect many different nodes than its neighbors.

**KEYWORDS: Connected Dominating Set, Wireless Sensor Networks, Minimum Weighted Connected Dominating Set, Minimal Weighted Connected Dominating Set, Minimal Routing Cost Connected Dominating Set**

# İÇİNDEKİLER

**Sayfa**

# ŞEKİL LİSTESİ

**Sayfa**

# TABLO LİSTESİ

**Sayfa**

# SÖZLÜK

| | | |
|---|---|---|
| **Node** | **:** | Düğüm |
| **Mote** | **:** | Telsiz duyarga düğümü |
| **2-Kenar** | **:** | Çizge içerisinde bir düğümün normalde komşusu olmayan ama komşularının komşusu olan düğümler |
| **IDLE** | **:** | Henüz baskın olmayan veya kapsanmayan düğüm durumu |
| **DTOR** | **:** | Baskın düğüm durumu |
| **DTEE** | **:** | Kapsanan düğüm durumu |
| **Ad hoc** | **:** | Herhangi bir kablosuz erişim noktası veya yönlendirici kullanılmadan iki veya daha fazla bilgisayarların kendi aralarında oluşturduğu ağ |
| **CONGEST** | **:** | Mesaj boyutlarının ağdaki düğüm sayısının logaritması ile sınırlandırılmış ağ modeli |
| **Approximation ratio:** | | Bir algoritmanın optimum sonuca ne kadar yakın sonuçlar ürettiğini gösteren oran |

# KISALTMALAR LİSTESİ

| | | |
|---|---|---|
| **TDA** | **:** | Telsiz Duyarga Ağları |
| **CDS** | **:** | Bağlı Baskın Küme |
| **MCDS** | **:** | Minimal Bağlı Baskın Küme |
| **MWCDS** | **:** | Minimal Ağırlıklı Bağlı Baskın Küme |
| **MOC-CDS** | **:** | Minimum Yönlendirme Maliyetli Bağlı Baskın Küme |
| **MWOC-CDS** | **:** | Minimal Ağırlıklı ve Yönlendirme Maliyetli Bağlı Baskın Küme |
| **CN-MWCDS** | **:** | Kritik Düğüm Tabanlı Minimal Ağırlıklı Bağlı Baskın Küme |
| **STA** | **:** | Steiner Ağacı Algoritması |
| **MIS** | **:** | Maksimal Bağımsız Küme |
| **MST** | **:** | Minimum Tarama Ağacı |
| **UDG** | **:** | Birim Disk Çizge |

# ÖNSÖZ

Yüksek lisans eğitimimde ve tez aşamasında beni yönlendiren, bilgilerini ve zamanını esirgemeyen tez danışmanım sayın Dr. Öğr. Ü. Elif HAYTAOĞLU'na sonsuz teşekkür ederim.

Yüksek lisans eğitimim süresince her konuda yardımlarını esirgemeyen çalışma arkadaşlarıma teşekkür ederim.

Bugüne kadar maddi manevi hiçbir desteğini esirgemeyen arkadaşlarıma ve kıymetli aileme teşekkürü bir borç bilirim.

# 1. GİRİŞ

Mühendislik uygulamalarında daha yüksek performanslı bilgisayarlara ihtiyaç her geçen gün artmaktadır. Yüksek performanslı sistemler işlemci saat frekansının arttırılması veya donanımın paralel bir şekilde kullanılmasıyla elde edilebilir. İşlemci saat frekansının artırılması bir noktadan sonra ya ekonomik nedenlerle ya da fiziksel sınırlar nedeniyle mümkün olmamaktadır. Bu nedenle daha fazla donanımın birlikte koordine bir şekilde kullanılmasıyla performans artırılması en çok tercih edilen yoldur. Farklı cihazlardaki donanım ve yazılım bileşenleri arasında haberleşme ve koordinasyonun mesajlaşma yoluyla sağlanabildiği sistemlere dağıtık sistem denir. Dağıtık sistemler günümüzde çok yaygın kullanılmaktadır. Bunlara örnek olarak internet, e-posta, ftp, telsiz duyarga ağları vs. gösterilebilir.

Telsiz duyarga ağları genellikle insan erişiminin ve uzaktan erişimin maliyetli olduğu durumlarda ortamdaki fiziksel koşulları dağıtık bir şekilde ölçmek ve işlemek için kullanılır. Kullanım alanı temel olarak askeri alanlar olsa da son zamanlarda sağlık, gıda, tarım vb. endüstriyel alanlarda da yoğun bir şekilde kullanılmaktadır.

Telsiz duyarga ağları, telsiz bir şekilde birbirlerine bağlanmış duyarga düğümlerinden (*node*) oluşur. Bu düğümler temel olarak gerekli hesaplamaları gerçekleştirebilen düşük hızlı işlemciler, çevredeki fiziksel koşulları ölçebilen duyargalar, kendi aralarında haberleşmeyi sağlayan alıcı-vericiler ve tüm bu bileşenlere enerji sağlayan enerji kaynaklarından oluşmaktadır. Şekil 1.1'de örnek bir telsiz duyarga düğümü gösterilmiştir. Telsiz duyarga ağları genellikle erişimi zor ortamlarda kullanıldığı için, bu enerji kaynakları da genellikle kısıtlıdır. Enerji kaynakları çoğunlukla pillerden oluşur ve bazen bu pilleri şarj eden güneş panelleri bulunabilir. Bu yüzden düğümlerin enerjileri bitebilmekte ve bu nedenle düğümler devre dışı kalmaktadır.

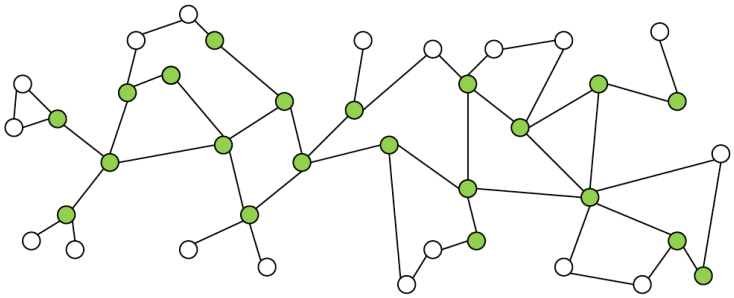
Şekil 1.1: MTM-CM5000-SMA telsiz duyarga düğümü (ADVANTIC SISTEMAS, 2018).

Cihazların enerji kaynakları bittiğinde bu kaynakları değiştirmek ya da şarj etmek sıkıntılı olabildiği gibi enerji kaynaklarının bitmiş olduğu andan itibaren sistemin çalışabilirliğini sürdürmek önemli bir problemdir. Dolayısıyla bu düğümlerde enerjiyi verimli kullanmak çok önemlidir. Düğümlerin enerjisi verimli kullanıldığında sistemin çalışabilirlik süresi en yüksek seviyede tutulabilmektedir.

Telsiz duyarga düğümlerinin kullanıldığı uygulamalarda genel olarak düğümler çevresinin fiziksel özelliklerini ölçüp işleyerek sonuçlarını baz düğüme göndermektedir. Bu düğümlerin ağ yapısı radyo kanallarından oluştuğu için çoğu düğüm baz düğümün kapsamı içerisinde değildir. Dolayısıyla baz düğüme gönderilecek mesajların baz düğüme ulaşması için aradaki başka düğümlerden yardım istenmektedir. Enerjinin büyük bir kısmı düğümler arası bahsedilen iletişim esnasında tüketilmektedir. Bu tüketimi düşürmek için çeşitli yöntemler kullanılmaktadır. Bunların başında iletişim için kullanılan yolların önceden belirlenmiş bir omurga üzerinden gerçekleştirilmesi gelmektedir. Bu omurgayı oluşturma yöntemlerinden biri, ağdaki bağlı baskın kümenin (CDS) bulunması ile gerçekleştirilmektedir.

Bağlı baskın kümenin bulunması için ağı oluşturan düğümlerin öyle bir alt kümesi oluşturulmalıdır ki bu düğümler ya bu kümenin içinde olmalıdır ya da bu alt kümedeki düğümlerden herhangi birine komşu olmalıdır. Şekil 1.2'de örnek bir bağlı baskın küme gösterilmiştir. Sistemdeki iletişim özellikle seçilmiş bağlı baskın kümenin içerisindeki düğümler üzerinden gerçekleştirilir. Bu seçilen düğümler iletişimde bir yol olarak kullanıldıkça enerjileri azalacak ve enerjileri bittiğinde sistemi çalışmaz hale getirecektir. Dolayısıyla bağlı baskın küme içerisindeki

düğümlerin enerjileri ne kadar yüksek ise sistemin yaşam süreside o kadar uzayacaktır.



Şekil 1.2: Bağlı baskın küme örneği (Ghaffari, 2014).

Bağlı baskın kümeyi oluşturan düğümler seçilirken düğümlerin ağırlıkları göz önünde bulundurulmalıdır. Ağırlıklar düğümlerin kalan enerjileri ile ters orantılı olarak ilişkilendirilmektedir. Bağlı baskın kümeyi oluşturacak düğümler seçilirken düğümlerin toplam ağırlıklarının ve sayılarının minimuma yakın olacak şekilde seçilmesi gerekmektedir. Ancak bağlı baskın küme içerisinde minimum sayıda ve/veya toplam minimum maliyetli cihaz bulunması problemi NP-tam bir problemdir (Clark ve diğ. 1991). Yani bu problem için polinom sürede çalışması garanti edilmiş bir algoritma henüz bulunmamaktadır.

Bu tez kapsamında bağlı baskın küme bulma problemi için kabul edilebilir sürelerde çalışan iki yeni algoritma geliştirilmiştir. İlk algoritmada Ding ve diğ. (2010)'in geliştirdiği MOC-CDS algoritmasından yola çıkılarak geliştirilen Minimal Ağırlıklı ve Yönlendirme Maliyetli Bağlı Baskın Küme algoritmasıdır. Bu algoritmada toplam ağırlığın minimize edilmesinin yanında bağlı baskın küme üzerinden gerçekleştirilecek olan iletişim yolları da kısaltılmıştır. İkinci algoritma ise ilk fazında baskın kümeyi oluşturan, ikinci fazında oluşturulan baskın kümeyi bağlı baskın kümeye dönüştüren Minimal Ağırlıklı Bağlı Baskın Küme algoritmasıdır. Bu yeni algoritmalarda toplamda kullanılan zaman ve mesaj maliyetleri azaltılmış ve oluşturulan bağlı baskın kümelerin düğüm sayıları ve toplam ağırlıkları azaltılmıştır.

Çalışmanın ilk bölümünde çalışma ile ilgili geniş özet verilmiş, ikinci bölümde literatürde yapılan çalışmalara yer verilmiş, üçüncü bölümde telsiz duyarga ağları önemli noktalarıyla anlatılmış, dördüncü bölümde bağlı baskın küme problemi detaylı bir şekilde anlatılmış, beşinci kısımda MOC-CDS algoritması ve tez kapsamında geliştirilen MWOC-CDS algoritması anlatılmış, altıncı kısımda Wang ve diğ. (2006)'nin geliştirdiği WANG algoritması, Dagdeviren ve diğ. (2015)'nin geliştirdiği ASYNSET ve ASYNTREE algoritmalarıyla tez kapsamında geliştirilen CN-WCDS algoritması anlatılmış ve son bölümde sonuçlar değerlendirilerek öneriler verilmiştir.

# 2. LİTERATÜR

Guha ve Khuller (1998) yaptıkları çalışmada CDS oluşturmak için iki merkezi açgözlü algoritma önermiştir. İlk algoritma tek fazlıdır. Algoritmanın başlangıcında tüm düğümler beyazdır. İlk olarak en fazla beyaz komşusu olan düğüm seçilir ve kendisi siyaha komşuları da griye boyanır. İlk turdan sonraki tüm turlarda gri ve gri düğümlerin komşuları içerisinden en fazla beyaz komşusu olan düğüm seçilir. Eğer seçilen düğüm beyaz ise o düğümü siyah düğüme bağlayan gri düğümde seçilir. Çizgede beyaz düğüm kalmayıncaya kadar turlar tekrarlanır ve CDS oluşturulur. İkinci algoritma ise çift fazlıdır. İlk fazında her turda en fazla beyaz komşusu olan düğüm DTOR olarak seçilir ve beyaz düğüm kalmayıncaya kadar devam eder. İkinci fazında ilk fazda seçilen düğümleri birleştiren düğümler Steiner Ağacı algoritması ile birleştirilir.

Das ve Bharghavan (1997) ad hoc ağlar üzerinde yönlendirme çalışmalarında Guha ve Khuller (1998)'ın algoritmalarını dağıtık sisteme uyarlayacak algoritmalar geliştirmişlerdir.

Cheng ve Ding (2004) çalışmalarında dört fazlı merkezi bir CDS algoritması önermişlerdir. Algoritmanın ilk fazında MIS oluşturulmuştur ve oluşturulan MIS içerisindeki düğümler kırmızıya boyanmıştır. İkinci fazda en çok kırmızı komşusu olan düğümler siyaha boyanarak DTOR seçilmiştir. Üçüncü fazda hala beyaz kalan düğümler için MST algoritması çalıştırılmış ve geriye sadece siyah ve gri düğümler kalmıştır. Algoritmanın son fazında birbirlerine bağlı olmayan siyah düğümleri bağlayan gri düğümler DTOR seçilerek CDS oluşturulmuştur.

Butenko ve diğ. (2004) merkezi ve budama tabanlı merkezi bir algoritma önermişlerdir. Algoritma başlarken tüm düğümler CDS'in içerisine dahil edilmiştir. Algoritmanın sonraki adımlarında gereksiz düğümler çıkartılarak düğüm sayısı minimuma indirilmeye çalışılmıştır.

Alzoubi ve diğ. (2002) iki fazlı ve MIS tabanlı dağıtık bir MWCDS oluşturma algoritması önermişlerdir. Algoritma ilk olarak MST oluşturarak en küçük

ağırlıklı düğümü DTOR olarak seçmektedir. DTOR olarak seçilen düğüm DTOR olduğunu gösteren bir mesajı broadcast etmektedir. Bu mesajı alan düğüm kendini DTEE yapmakta ve DTEE olduğunu gösteren bir mesajı broadcast etmektedir. Kendinden küçük ağırlığa sahip olan tüm komşularından DTEE mesajı alan düğüm kendini DTOR ilan etmekte ve algoritma bu şekilde tüm düğümler DTOR veya DTEE olana kadar devam etmektedir. Algoritmanın ikinci fazında DTOR düğümlerini birleştiren düğümler seçilmekte ve CDS tamamlanmaktadır.

Cardei ve diğ. (2002) çalışmalarında Alzoubi ve diğ. (2002)'nin algoritmasını MST oluşturmadan gerçekleştiren bir algoritma geliştirmişlerdir. Algoritmada MST yerine düğümler için aktif parametresi eklenmiştir. Başlangıçta tüm düğümler pasiftir. Lider düğüm DTOR mesajını göndererek başlamaktadır. DTOR mesajını alan düğümler kendilerini DTEE yapar, aktifleşir ve DTEE mesajı gönderir. DTEE mesajı alan beyaz düğümler de kendilerini aktif hale getirirler. Tüm aktif komşuları içerisinde en küçük ağırlığa sahip olan düğüm DTOR olmaktadır ve bu döngü beyaz düğüm kalmayıncaya kadar devam etmektedir.

Alzoubi ve diğ. (2002) çalışmalarında iki fazlı ve dağıtık bir algoritma önermişlerdir. Bu algoritmanın ilk fazında beyaz komşuları içerisinde ID'si en küçük olan düğüm kendini DTOR olarak ilan etmekte ve beyaz komşuları DTEE olmaktadır. Beyaz düğüm kalmayıncaya kadar algoritma bu şekilde devam etmektedir. İkinci fazda her DTOR düğüm 3-kenar uzaklığındaki diğer DTOR düğümler arasındaki yolları hesaplar ve bu yollardaki düğümler kendilerini CDS içerisine dahil ederek CDS oluşturulmaktadır.

Wu ve Li (1999) ve F Dai ve Wu (2004) yaptıkları çalışmalarda budama tabanlı merkezi ve dağıtık algoritmalar önermişlerdir. Bu algoritmalar iki fazlıdır. İlk fazda her turda iki beyaz komşusu olan tüm düğümler DTOR olarak seçilmektedir. Çizgede beyaz düğüm kalmayınca, ilk faz içinde gereksiz düğümlerinde olduğu fazla sayıda düğümü içinde bulunduran bir CDS oluşturmaktadır. Bu yüzden ikinci fazda, ilk fazda oluşturulan CDS içerisindeki düğülerin bir kısmı budanmaktadır. Budama için iki kural bulunmaktadır. Birincisi bir DTOR düğümün kapsadığı tüm DTEE düğümler kendisinden daha büyük bir ID'li DTOR tarafından kapsanmalıdır. İkincisi düğümün komşusu olduğu tüm DTOR düğümlerin budanacak düğümün dışında bir

DTOR düğüme daha komşu olmalıdır. Bu iki kurala uyan tüm DTOR düğümler DTEE düğüme çevrilir ve MCDS oluşturulur.

Mohanty ve diğ. (2017) 2-kenar komşuluk bilgisini kullanarak dağıtık bir minimum CDS bulma algoritması önermişlerdir. Önerilen algoritma üç fazlıdır. İlk fazda Minimum-Maksimal Bağımsız Küme bulunmaktadır ve bu kümeye sahte DS denmektedir. İkinci fazda, bulunan sahte DS içerisindeki düğümleri birbirlerine bağlamak için Steiner Ağacı algoritması kullanılarak sahte CDS bulunmuştur. Üçüncü fazda ise oluşturulan sahte CDS içerisindeki gereksiz düğümler çıkartılarak minimum CDS oluşturulması hedeflenmiştir.

Ghaffari (2014) çalışmasında düğümlerin her turda maksimum mesaj boyutunun altında mesaj gönderdiği minimum ağırlıklı CDS algoritması önermiştir. Algoritma düğümlerin gönderdiği mesaj boyutlarının küçük tutulduğu CONGEST modele göre tasarlanmıştır.

Misra ve Mandal (2010) yaptıkları çalışmada MIS tabanlı sezgisel ve dağıtık bir CDS oluşturma algoritması önermişlerdir. Önerilen algoritmada DTOR düğümler seçilirken kapsama oranına göre seçilmektedir. Kapsama oranı kapsanan düğümlerin kapsayan düğümlere oranıdır. Kapsama oranı en yüksek olan düğümler DTOR olarak seçilmektedir. DTOR ve DTEE düğümler seçildikten sonra DTOR düğümleri birbirlerine bağlamak için Steiner Ağacı algoritması kullanılmıştır.

Fei Dai ve Wu (2006) k-Bağlı k-Baskın CDS oluşturma algoritması üzerine çalışmışlardır. Önerilen algoritmanın çalıştırılması sonucunda klasik CDS yerine hata toleransı ve ağ esnekliği daha yüksek olan k-Bağlı k-Baskın CDS oluşturulmuştur. Bu sayede CDS içerisindeki düğümlerden herhangi biri veya birkaçının bozulması durumda sistem yaşamını devam ettirebilecektir.

Bir çizge için birden fazla CDS bulunabilmektedir. Bulunan CDS sayısı ne kadar fazla ise ağın yaşam süreside o denli artırılabilir. Çizge içerisinde bulunan CDS'lerin ne zaman ve hangi sırayla kullanılacağı bir problemdir. Misra ve Mandal (2009) yaptıkları çalışmada çizge içerisindeki CDS'lerin nasıl en iyi kullanılabileceği üzerine çalışmışlardır.

Shi ve diğ. (2017) kendi enerjisini kısmen üretebilen düğümlerden oluşan ağlarda maksimum sayıda CDS bulmak için bir algoritma geliştirmişlerdir. Bu çalışmanın diğer çalışmalardan en büyük farkı düğümlerin enerjileri sabit değil sürekli artan ve azalan durumda olabilmeleridir.

Ding ve diğ. (2010) CDS'lerin yönlendirme maliyetlerinin minimuma indirilmesi üzerinde çalışmışlardır. Yönlendirme maliyetini minimuma indiren CDS problemini oluşturup MOC-CDS ismini vermişlerdir. MOC-CDS probleminde bir çizge içerisinde herhangi iki düğüm arasında bulunan en kısa yolların CDS içerisinde olması gerekmektedir. Ding ve diğ. bu problemi çözmek için iki kenar uzaklıktaki tüm düğümlerin aralarındaki düğümleri CDS içerisine koymanın, tüm düğüm ikililerinin aralarındaki en kısa yolu CDS içerisine dahil etmiş olacağını kanıtlamışlardır. Bu yüzden her turda düğümler aralarında bağlantı kurdukları düğüm sayılarına göre DTOR seçilmektedirler. Birbirine bağlanmayan 2-kenar uzaklığı bulunan düğüm ikilisi kalmayıncaya kadar algoritma devam etmektedir. Son oluşan DTOR düğümleri bir CDS oluşturmaktadır.

Wang ve diğ. (2006) açgözlü CDS oluşturma algoritmalarının yeterli olmadığını öne sürmüş ve bu algoritmalara sezgisel bir yöntem ekleyerek yeni bir dağıtık MWCDS oluşturma algoritması önermişlerdir. Algoritma üç fazdan oluşmaktadır. İlk fazın başlangıcında tüm düğümler IDLE, beyazdır ve komşularının ağırlıklarını bilmektedir. Eğer bir düğümün ağırlığı beyaz komşularının ağırlıkları içerisinde en küçüğü ise 2-kenar komşuluk listesini toplamakta ve bu listede yerel bir MIS benzeri algoritma çalıştırmaktadır. Bu algoritma sonucu ya kendisini ya da 2-kenar uzaklığındaki düğüm ya da düğümleri DTOR olarak seçmektedir. Çizgede beyaz düğüm kalmayıncaya kadar aynı işlemler tekrarlanmaktadır ve ardından birinci faz sona ermektedir. İkinci fazda DTOR düğümler kendi aralarındaki yolları bularak sadece DTOR düğümlerin köşe olduğu aralarındaki yolları oluşturan düğümlerin toplam ağırlıklarının kenar ağırlıkları olduğu sanal bir çizge oluştururlar. Üçüncü fazda oluşturulan bu sanal çizge üzerinde MST algoritması çalıştırılır, seçilen sanal kenarlar üzerindeki düğümler DTOR olarak seçilir ve CDS oluşturulmuş olur.

Dagdeviren ve diğ. (2015) TDA'larda enerjinin önemini vurgulayıp düğümlerin ağırlıklarını kalan enerjilerinin ters orantısı şeklinde hesaplamışlardır. Önerdikleri algoritmada düğümler arası eşlemeyi sağlamak için senkronizasyon

ağacı kullanılmaktadır. Algoritmada iki fazdan oluşmaktadır ve dağıtık bir MWCDS algoritmasıdır. İlk fazda DTOR düğümler seçilirken düğümün 2-kenar uzaklıktaki komşularının içerisinde en düşük ağırlık oranına sahip olması gerekmektedir. Düğümlerin ağırlık oranı kendi ağırlıklarının, beyaz komşularının ağırlıklarının toplamına oranıdır. Her turda seçim işlemleri tekrarlanır ve beyaz düğüm kalmayıncaya kadar devam eder. İkinci fazın başlangıcında birbirine komşu olan DTOR düğümlerin ağaç oluşturmaları için Gallager ve diğ. (1983)'nin geliştirdiği MST algoritması kullanılarak küçük DTOR ağaçları oluşturulmaktadır. Ardından oluşan bu küçük ağaçları bir bütün haline getirebilmek için geliştirilmiş bir Steiner Ağacı algoritması kullanılmaktadır.

# 3. TELSİZ DUYARGA AĞLARI

Yeni nesil telsiz haberleşme sistemleri felaket yönetimi, arama kurtarma, acil durumlar ve askeri ağlarda kullanılmaktadır. Bu hayatta kalabilir, dinamik ve verimli olmalıdır. Fakat bu özelliklere sahip olan ağların merkezi ve yönetilen ağlar olması mümkün değildir (Li, 2008). Çünkü ağ topolojisinin sürekli ve sabit olmaması, tüm ağ hareketliliğinin ve topolojinin merkezi olarak yönetilmesini zorlaştırmaktadır. Çevresindeki topolojiyi keşfetmek te dahil olmak üzere tüm ağ hareketliliğini düğümler kendi başlarına yönetmelidir. Bu tür herhangi bir yönetme merkezi olmadan birbirleri ile mesajlaşarak koordine bir şekilde çalışan sistemlere dağıtık sistemler denir .

Endüstriyel alanların çoğunda ortamı izlemeye ve ortamdaki fiziksel verileri toplamaya ihtiyaç duyulmaktadır. Fiziksel verileri ölçmek için duyargalar kullanılır. Duyargalardan verileri alabilmek için genelde kablolar kullanılır. Fakat kablolar her türlü ortama yerleştirilemedikleri için kablolu duyarga kullanmak bazen pahalıya mal olmaktadır. Böyle durumlarda telsiz duyargalar bir ana bilgisayara bağlanılarak kullanılabilir. Ancak kullanılan merkezi bilgisayarın konumu tüm telsiz duyargaların iletişim alanında bulunamadığı durumlarda, merkezi yönetime ihtiyaç duymayan rastgele yerleştirilmiş telsiz duyarga düğümlerinin ortamdaki veriyi ölçüp işleyip kendi aralarında kablosuz mesajlaşarak ana bilgisayara iletebilen telsiz duyarga ağları (WSN) kullanılmaktadır. Telsiz duyarga ağları konusu, birçok teknoloji, algoritma ve ürünü bir arada kullandığı için fazlaca karmaşık bir konudur.

## 3.1 TDA'ların Tarihi

Çoğu gelişmiş teknolojide olduğu gibi telsiz duyarga ağlarının başlangıcı da askeri ve ağır endüstrilere dayanmaktadır. Telsiz duyarga ağlarının ilk örneği, Amerika Birleşik Devletleri tarafından geliştirilmiş dağıtık bir ses gözetleme sistemi olan SOSUS'tur. SOSUS 1950'lerde Atlantik ve Pasifik okyanuslarında Sovyet denizaltılarını keşfetmek ve izlemek için geliştirilmiştir. Bu sistem okyanus dibinde stratejik önem taşıyan bölgelere yerleştirilmiş sualtı mikrofonlarından alınan verileri

dağıtık bir şekilde işleyerek çalışmaktadır. SOSUS günümüzde ise sualtı vahşi yaşamı ve volkanik hareketliliği izlemek için kullanılmaktadır (Chong & Kumar, 2003).

Duyarga ağlarındaki yeni nesil çalışmalar 1980'lerde İleri Savunma Araştırma Proje Ajansı (DARPA) tarafından Dağıtık Duyarga Ağları (DSN) programı ile başlamıştır. Dağıtık duyarga ağ mimarisinin başarısından esinlenilerek, üniversitelere ve enstitülere yerleştirilen 200 civarı sunucu ile başlayan ARPANET daha sonra TCP/IP protokolünün geliştirilmesiyle internetin temellerini oluşturmuştur (Chong & Kumar, 2003).

DSN cihazlarında kullanılan bileşenler 1978'de düzenlenen Dağıtık Duyarga Ağları Çalıştay'ında tanımlanmıştır. Tanımlanan bileşenler duyargalar, haberleşme bileşenleri ve işlemcilerden oluşmaktadır (Carnegie-Mellon Univ. Pittsburgh, 1978).

MIT ve Cambridge'teki araştırmacılar, dağıtılmış mikrofonlar üzerinde sinyal özetleme ve eşleştirme tekniklerini kullanarak helikopterleri izlemek için bilgi-tabanlı sinyal işleme üzerinde çalışmışlardır (Myers & Oppenheim, 1984).

Son zamanlardaki işlemci ve haberleşme teknolojilerindeki iyileşmeler telsiz duyarga ağlarına hızlı bir ivme kazandırmıştır. Daha az enerjiyle daha hızlı çalışan işlemciler, boyutları küçülen duyargalar ve telsiz haberleşme teknolojilerinin yaygınlaşması sonucunda telsiz duyarga ağları araştırmaları yaygınlaşmıştır. Bu gelişmelerin ardından DARPA Duyarga Bilgi Teknolojisi (SensIT) programını başlatmıştır (Kumar & Shepherd, 2001).

## 3.2 TDA'ların Donanım Yapısı

TDA'ları oluşturan düğümlere duyarga düğümleri veya *mote* denir. *Mote*'ların donanım yapısını genel olarak inceleyecek olursak dört alt sistemden oluşmaktadır (Healy ve diğ. 2008). Bu sistemler hesaplama sistemi, haberleşme sistemi, duyarga sistemi ve güç sistemleridir.

Hesaplama sistemi mote üzerindeki tüm bileşenlerin birlikte çalışmasını yöneten sistemdir. Tüm bilgi teknolojisi cihazlarında olduğu gibi bu sistem işlemci

ve belleklerden oluşur. Bir TDA düğümü için işlemcinin düşük enerji tüketmesi gerekmektedir. Özellikle uyku durumunda olan bir düğümün işlemcisinin enerjiyi çok düşük tüketmesi gerekmektedir. Çünkü çoğu TDA uygulamasında bir TDA düğümü çalışma zamanının %95'ini uyku durumunda geçirmektedir (Healy ve diğ. 2008). Bellek olarak duyarga düğümlerinde sadece RAM bulunmaktadır. Harici belleğe yazmak için kullanılacak enerji, veriyi işleyip mesajlaşarak ana bilgisayara göndereceği enerjiden daha fazla olduğu için duyarga düğümlerinin çoğunda harici bellek bulunmamaktadır.

Haberleşme sistemi duyarga düğümlerinin kendi aralarında ve baz bilgisayarla veri alışverişini gerçekleştirdikleri genellikle telsiz sistemlerdir. Telsiz haberleşme için kullanılan teknolojiler genellikle kızılötesi, RFID ve radyo sinyalleridir. Kızılötesi haberleşme sistemi düşük enerji ile çalışabilmesi ve ucuz olmasına rağmen, iki cihaz arasına engel girdiğinde bozulduğu veya iki cihaz aralarında engel olmayacak şekilde yerleştirilemediği için TDA'lara uygun değildir (Healy ve diğ. 2008). RFID çok geniş bir şekilde kullanılsa da, yüksek enerji ihtiyaçları ve kısa menzillerinden dolayı TDA'larda kullanılmazlar. TDA'lar için en uygun haberleşme teknolojisi radyo sinyalleridir. Çünkü hem enerji maliyeti olarak hem de menzil uzunluğu olarak TDA'ların ihtiyaçlarına cevap verebilmektedir. Birçok duyarga düğümü haberleşme altyapısı olarak IEEE 802.15.4 standartlarına uygun radyo alıcı-vericilerini kullanmaktadır.

Güç sistemi duyarga düğümündeki tüm bileşenlerin enerji ihtiyaçlarını karşılamaya yarayan enerji kaynaklarıdır. Genellikle sürekli enerji kaynağına erişimleri olmadığından veya çok maliyetli olmasından dolayı bataryalardan oluşur ve bazıları şarj edilebilmektedir. Bataryalar TDA'ların yaşam süresini belirleyen bileşenlerdir. Bu yüzden bataryalardaki enerjinin verimli kullanılması gerekmektedir. Duyarga sistemi ise çevredeki fiziksel verilerin elektrik sinyallerine çevirip ölçen sistemdir. Duyargalardan alınan bilgiler diğer bileşenler aracılığıyla işlenmektedir.

## 3.3 TDA Örnekleri

TDA'lar günümüzde birçok alanda kullanılmaktadır. Bu alanların en başında ise çevre gözetleme/izleme sistemleridir. Çevre izleme sistemleri belirli bir bölgeye yerleştiren duyargaların o çevredeki fiziksel verileri ölçüp değerlendirmesiyle çalışır. Bu sistemlerin örnekleri en çok askeri alanda görülmektedir ve genellikle düşman tehdidini önceden belirlemek için kullanılır. Çevre izleme sistemleri ticari alanlarda da kullanılır. Örneğin petrol ve doğalgaz boru hatlarındaki sızıntıların tespit etmek için kullanılır.

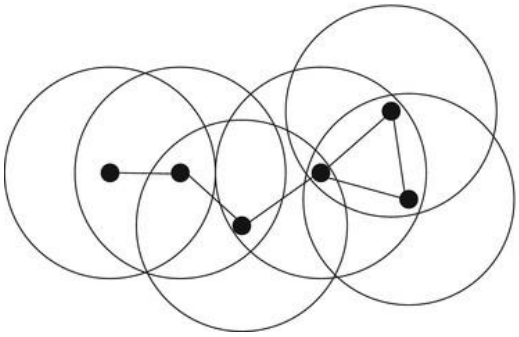
TDA'lar son zamanlarda sağlık alanlarında da kullanılmaya başlanmıştır. Vücuda yerleştirilen birtakım duyargalar birbirleri arasında haberleşerek verileri analiz edebilmektedir. Yapılan bu analiz sonucunda hastalık teşhisi veya tedavi takibi yapılabilmektedir. O'Donovan ve diğ. (2009) yaşlı insanların vücutlarına ivme, tansiyon ve nabız ölçen duyargalar yerleştirerek hareketlerinin analizlerini ve düşme durumu değerlendirmelerini yapmışlardır.

TDA'lar çevresel sorunların çözülmesinde büyük bir rol oynamaktadır. Orman yangını, hava ve su kirliliği gibi çevresel sorunlarda ve toprak kayması gibi doğal felaketlerde kullanılmaktadır. Örneğin bir orman yangını sırasında yangının ne tarafa doğru ilerleyeceği duyargalar tarafından ölçülen verilerin işlenmesiyle kısa sürede tespit edilebilir ve müdahalenin doğru zamanda doğru bölgeye yapılmasını hızlandırabilir. Şehirlerde zehirli gaz salınımını tespit edip bölgede yaşayan nüfusun bölgeyi boşaltması sağlanarak insanların hayatları kurtulabilir.

Askeri, çevresel ve sağlık alanların dışında TDA'lar ticari ve tarım alanlarında da kullanılmaktadır. Tarım için kullanılan bir arazinin sıcaklık ve nem gibi değerleri ölçülüp hangi bölgenin ne sıklıkla ve ne kadar sulanması gerektiği canlı olarak hesaplanıp arazinin daha verimli kullanılmasına olanak sağlamaktadır. Ticari örneklerine bakıldığında fabrikalardaki makinelerin sürdürülebilir şekilde çalışmalarını sağlamak amacıyla durum tabanlı yönetim için TDA'lar kullanılmaktadır (Tiwari, ve diğ. 2007). Aynı şekilde veri merkezleri içerisine yerleştirilen duyargalar sayesinde, oluşabilecek kötü durumlar engellenebilmektedir. Fabrikalarda kullanılan TDA'lara bir diğer örnek ise Anastasi ve diğ. (2009) Sicilya şarabı üretimindeki kaliteyi artırmak için TDA kullanmışlardır.

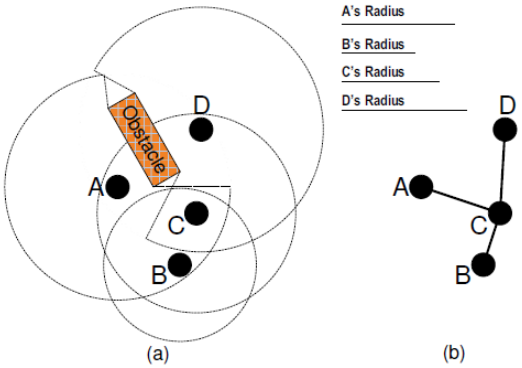
## 3.4 TDA İçin Ağ Modelleri

TDA'ların topolojilerinin gösterilebilmesi için çizgeler kullanılır. Kullanılan çizgeler genellikle disk çizgeleridir. Bu çizgelerin özelliği her düğümün bir menzili olması ve bu menzil içerisindeki düğümlere mesaj gönderebiliyor olmasıdır. Disk çizgelerinin içerisinde en çok kullanılanı ise birim disk çizgeleridir (UDG). Birim disk çizgeleri her düğümün eşit menzile sahip olduğu ve bu menzil içerisindeki diğer düğümlere hem mesaj gönderebildiği hem de bu düğümlerden mesaj alabildiği çizgelerdir (Clark ve diğ. 1991). Örnek bir birim disk çizgesi Şekil 3.1'de gösterilmiştir.



Şekil 3.1: Birim disk çizgesi örneği.

Bazı durumlarda yukarıdaki iki model yeterli olmayabilir. Örneğin Ding ve diğ. (2010) düğümlerin menzillerinin farklı olabileceğini ve aralarına engeller girebileceğini Şekil 3.2'de göstermiştir. Şekil 3.2'de normal şartlarda D düğümünün A düğümüne mesaj gönderebilecek menzile sahip olmasına rağmen aralarına giren bir engel nedeniyle bağlantı kurulamadığını göstermiştir.




Şekil 3.2: Ding ve diğ. (2010)'nin örnek ağ modeli.

## 3.5 TDA İçin İşletim Sistemleri

İşletim sistemi donanım kaynaklarını yöneten, bağlı cihazları kontrol eden ve uygulamalar ile donanım arasındaki bağlantıyı sağlayan yazılımdır. PC'ler için geliştirilmiş işletim sistemleri TDA'lar için uygun değildir. Çünkü TDA düğümlerinin kaynakları kısıtlıdır ve yeterli hafızaları olmadığı için veri odaklı çalışırlar. Bu yüzden TDA düğümleri için geliştirilmiş özel işletim sistemlerine ihtiyaç duyulmaktadır. Bu işletim sistemlerinin genel özellikleri az kaynak ile çalışabilmesi, düşük enerji tüketmesi, gerektiğinde sistemi uyutup uyandırabilmesi vb. özelliklerdir (Sohraby ve diğ. 2007). Günümüze kadar TDA'lar için çeşitli işletim sistemleri üretilmiştir. Bu işletim sistemlerinden bazıları TinyOS (P Levisve diğ. 2005), Contiki OS (Dunkels ve diğ. 2004), Magnet OS (Barr ve diğ. 2002), MANTIS (Abrach ve diğ. 2003) ve SenOs'tur (Hong ve Kim, 2003).

### 3.5.1 TinyOS

TinyOS TDA'lar için uygulamalara özel geliştirilmiş olay-tabanlı gömülü bir işletim sistemidir. University Of California, Berkeley, Intel Research ve Crossbow Technology tarafından geliştirilmeye başlanmıştır. Programlama dili olarak C dilinden geliştirilen, olaylar, bileşenler, ara yüzler ve görevlerden oluşan NesC (*Network embedded systems C*) dilini kullanmaktadır. TDA'larda en çok kullanılan işletim sistemidir (Levis ve diğ. 2005). Nesneye yönelik programlamayı desteklememektedir. Toplam kapladığı hafıza alanı 400 byte'tan daha azdır.

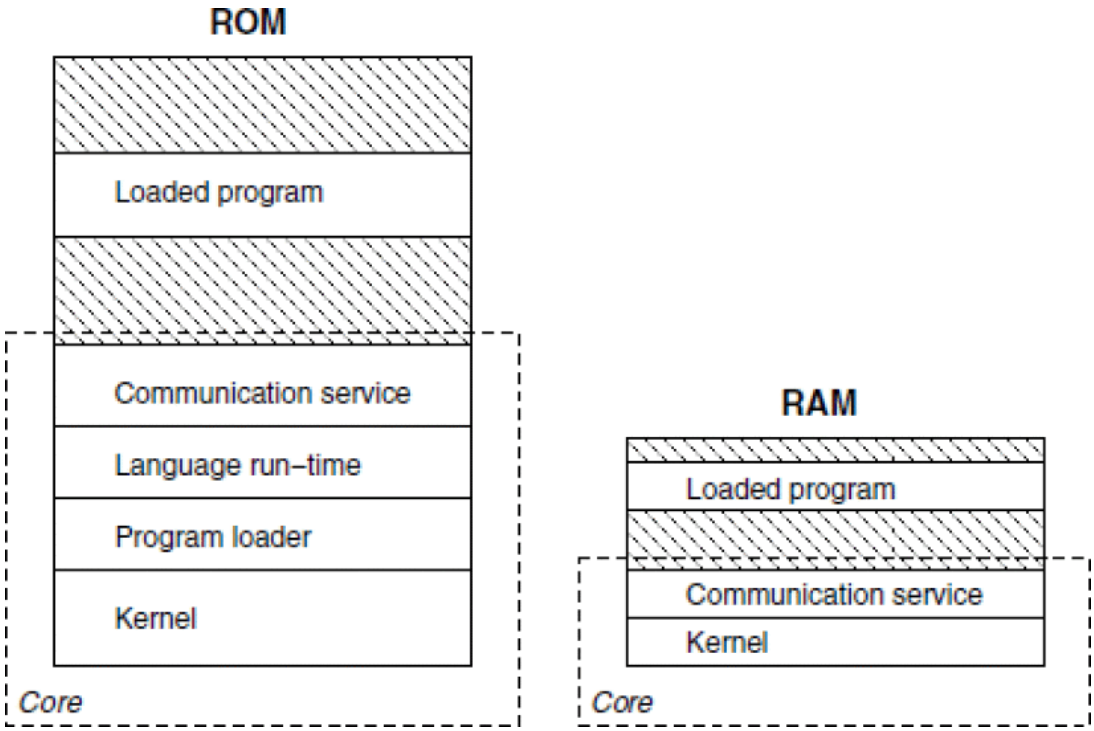
### 3.5.2 Lite OS

Lite OS Illinois Üniversitesi tarafından telsiz duyarga ağları için geliştirilmiş Unix benzeri bir işletim sistemidir (Farooq ve Kunz, 2011). Geliştirilmesinin amacı TDA'lar için Unix benzeri bir işletim sisteminin daha önce yapılmamış olmasıdır. Nesneye yönelik programlamaya olanak sağlamaktadır ve Unix benzeri bir kabuğa sahiptir. Çoklu iplik yapısını desteklemektedir ve karşılıklı dışlamayı sağlamak için atomic_start() ve atomic_end() fonksiyonları vardır. Öncelik tabanlı planlayıcı kullandığı ve işlemi bölmediği için gerçek zamanlı uygulamalar için uygun değildir.

Çünkü gerçek zamanlı çalışması gereken işlem çalışması gerektiği zamanda, çalışmakta olan işlemin uzun sürmesi durumunda çalışamayabilir (Farooq ve Kunz, 2011).

### 3.5.3 Contiki OS

Contiki OS nesnelerin interneti için geliştirilmiş açık kaynak bir işletim sistemidir. Sistemin çalışabilmesi için 10KB'dan daha az RAM ve 30KB'dan daha az ROM'a ihtiyaç duyulmaktadır. IPv4 ve IPv6 protokollerini desteklemektedir ve tamamen internet protokolü üzerinden çalışabilmektedir. IPv4 ve IPv6 yığınlarının yanı sıra daha az bant genişliğine ihtiyaç duyan Rime iletişim yığınını kullanabilmektedir. TDA'lar için geliştirilen diğer işletim sistemlerinin aksine statik değildir, uygulama çalışırken yeni programlar yüklenebilmektedir. Bu sayede çok katlı duyarga ağları üzerinde birden fazla program çalıştırabilmektedir (Dunkels ve diğ. 2004). Programlar için bellek bölüntüleme yapısı Şekil 3.3'te verilmiştir.




Şekil 3.3: Contiki OS programlar için bellek bölüntüleme yapısı (Dunkels ve diğ. 2004).

## 3.6 TDA İçin Benzetim Ortamları

Telsiz duyarga ağları çok fazla düğümün bir araya gelmesiyle oluşur. Bu düğümlerin maliyetleri yüksek olduğundan, geliştirme veya araştırma yapılırken sürekli kullanımları mümkün değildir. Elinizde yeterince TDA düğümü olsa bile bunlar üzerinde çalışmak zaman ve mekan açısından maliyetlidir. Çünkü yüzlerce veya binlerce düğümün kurulum işlemlerinin tamamlanması ve gerekli bölgeye yerleştirilmesi için çok büyük zamana ve mekana ihtiyaç vardır. Zaman ve mekan probleminin olmadığı durumlarda da geliştirilen uygulamanın hata yakalama işlemi çok zordur. Bu nedenlerden dolayı TDA'lar üzerinde yapılan araştırma geliştirme işlemlerinde benzetim ortamları kullanılmaktadır (Jevtić ve diğ. 2009).

Günümüze kadar TDA'lar için birçok benzetim ortamı geliştirilmiştir. Bunlardan bazıları NS2 (Mohapatra ve Kanungo, 2012), OMNeT++ (Varga ve Hornig, 2008), TOSSIM (Philip Levis ve diğ. 2003) ve COOJA'dır (Österlind ve diğ. 2006). NS2 ve OMNeT++ normalde bir ağ benzetim ortamı olarak geliştirilmeye başlanmıştır. Daha sonra TDA'ların popülerleşmesiyle TDA'lara uygun araçlar geliştirilerek TDA'lar için kullanılmaya başlanmışlardır (Mekni ve Moulin, 2008). Bu iki benzetim ortamı C++ ile yazılmış olup nesneye yönelik programlama destekleri vardır. NS2 ve OMNeT++ üzerinde geliştirilen uygulamalar gerçek alanlarda uygulanamamaktadır. Çünkü bu benzetim ortamları TDA'lar için geliştirilen bir işletim sistemini desteklememektedirler.

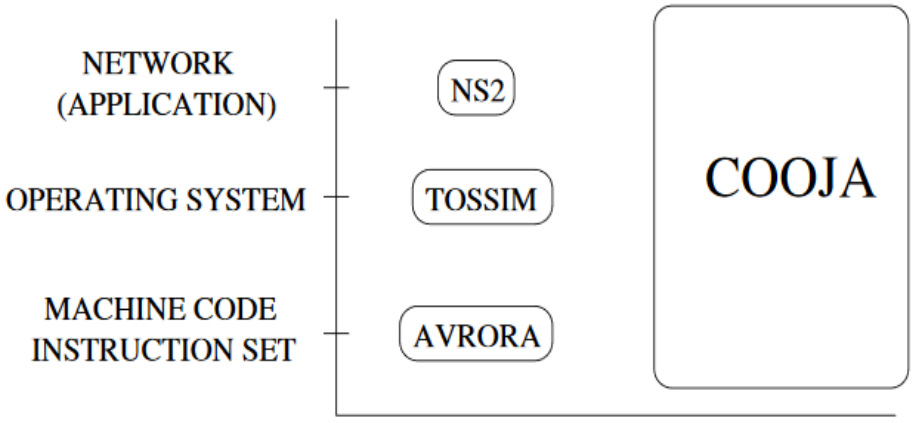
### 3.6.1 TOSSIM

TOSSIM (TinyOS Simulator) TinyOS işletim sistemini kullanan TDA uygulamaları için geliştirilmiş bir benzetim ortamıdır. TinyOS işletim sistemini geliştirenler tarafından geliştirilmiştir ve TOSSIM sayesinde TinyOS üzerindeki bazı hataları gidermişlerdir. TOSSIM bir benzetim ortamı kütüphanesidir. Benzetim ortamının çalışması için düğümlerin, aralarındaki bağlantıların ve gürültülerin kodlayarak ayarlanması gerekmektedir. TOSSIM Python ve C++ olmak üzere iki programlama dilinde programlanabilmektedir. Temel olarak bir görsel ara yüzü bulunmamaktadır fakat farklı kütüphaneler kullanılarak görsel ara yüz

eklenebilmektedir. Enerji ölçümü için kendi içerisinde bir kütüphane bulundurmamaktadır (Philip Levis ve diğ. 2003).

### 3.6.2 COOJA SIM

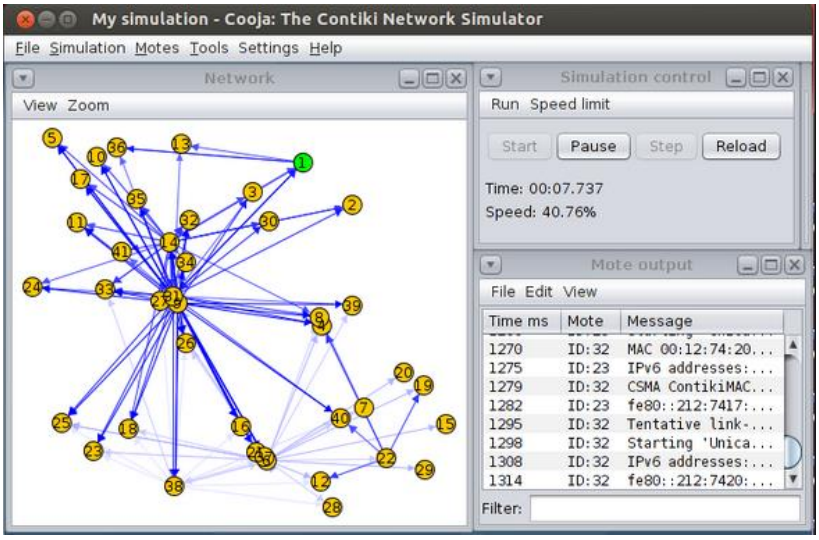
COOJA Java tabanlı Contiki OS kullanan düğümler için geliştirilmiş bir duyarga ağ benzetim ortamıdır. COOJA, farklı uygulamaları farklı cihazlar üzerinde aynı benzetim ortamında esnek bir şekilde test edilebilmeyi desteklemektedir. COOJA diğer benzetim ortamlarının aksine birçok katmanda benzetim sağlayabilmektedir (bkz. Şekil 3.4) (Österlind ve diğ. 2006). Örneğin: ağ katmanında düğümlerin kullandığı radyo aygıtlarını değiştirebilmekte, işletim sistemi katmanında düğümler üzerinde kullanıcı işlemlerini ve farklı programları çalıştırabilmekte ve makine kodu katmanında Contiki işletim sistemine ihtiyaç duymadan düğümler çalıştırılabilmektedir.




Şekil 3.4: COOJA ve rakiplerinin benzetim sağladığı katmanlar (Österlind ve diğ. 2006).

COOJA ile geliştirilen uygulamalar Contiki işletim sistemi üzerine yazılan programlar olduğu için gerçek ortamda da çalışabilmektedir. COOJA benzetim ortamının çalışması için csc uzantılı dosyalara topolojinin, cihazların ve programların xml ile kodlanması gerekmektedir.

COOJA dışarıdan bir kütüphaneye ihtiyaç duymadan görsel bir ara yüz ile çalışabilmektedir ve csc dosyalarını kodlama yapmadan bu ara yüz sayesinde oluşturmaya imkan sağlamaktadır. Şekil 3.5'te örnek bir ekran görüntüsü verilmiştir.




Şekil 3.5: COOJA ekran görüntüsü.

# 4. BAĞLI BASKIN KÜME(CDS) PROBLEMİ

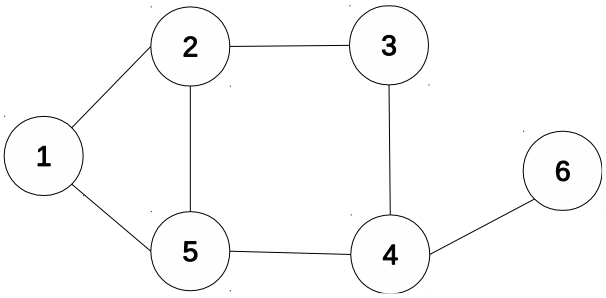
Bağlı baskın kümeler ilk olarak 1970'lerde çalışılmaya başlanmıştır. 1998'de iletişim ve bilgisayar ağları üzerinde uygulanması sonucunda popülerlik kazanmış ve günümüze kadar birçok kişi tarafından gerek teorik gerekse pratik olarak çalışılmıştır (Chinchuluun ve diğ. 2009) .

Telsiz duyarga ağları gibi telsiz ağlarda gerçek bir ağ altyapısı bulunmamaktadır. Bu ağlar genellikle bir veya birkaç tane baz bilgisayar ve birbirinden bağımsız olan daha ucuz işlemcileri olan düğümlerden oluşurlar. Ağ içerisindeki tüm düğümlerin, kapsama alanlarının yeterince büyük olmamasından dolayı baz bilgisayarlara ve komşu olmadığı diğer düğümlere doğrudan iletişimi bulunmamaktadır. Böyle durumlarda sanal bir ağ altyapısı kullanılmaktadır. Bu sanal ağ altyapısı bağlı baskın kümelerin en çok kullanıldığı problemdir. Yani birbirleri arasında doğrudan iletişimi olmayan düğümler, haberleşmek için sanal bir altyapı kullanmaktadır. Bu altyapı iletişim kuracak düğümler arasındaki düğümlerin sırayla kullanılmasıyla sağlanır. Hangi düğümlerin hangi sırayla kullanılarak mesaj iletişimine katkıda bulunacağını ise bağlı baskın kümeler kullanılarak oluşturulan omurga belirlemektedir (Du ve Pardalos, 2005).

CDS içerisindeki düğümlere baskın (DTOR), DTOR düğümlere komşu olan ve DTOR olmayan düğümlere kapsanan (DTEE) denmektedir.

## 4.1 Matematiksel Model

Telsiz duyarga ağları gibi dağıtık ağlar kağıt üzerinde bir çizge ile temsil edilmektedir. Çizgeler *G(V,E)* şeklinde gösterilmektedir. Buradaki *V* çizge içerisindeki tüm düğümleri içinde bulunduran köşe kümesi, *E* ise *V* kümesinin içerisindeki düğümlerin birbirleri ile olan bağlantılarının bulunduğu kenar kümesidir.




Şekil 4.1: Örnek bir çizge.

Baskın küme (*DS*) *V* kümesinin öyle bir alt kümesidir ki: *V* içerisinde bulunan tüm düğümler ya DS içerisinde olmalı ya da DS içerisindeki en az bir düğüm ile aralarında kenar olmalıdır. Örneğin Şekil 4.1'de gösterilen çizgede baskın küme olarak {2, 4} kümesi seçilebilir. Çünkü 2 ve 4 numaralı düğümler DS içerisindedir, 1, 3, 5 ve 6 numaralı düğümlerin DS içerisindeki 2 veya 4 numaralı düğümler ile arasında kenarları bulunmaktadır.

Bağlı baskın küme (CDS) ise baskın küme içerisinde bulunan düğümlerin birbirlerine bağlı olduğu kümelerdir. Örneğin Şekil 4.1'de gösterilen çizgede {2, 4} kümesi bir baskın küme olmasına rağmen bağlı baskın küme değildir. Bağlı baskın küme olabilmesi için küme içerisinde bulunan 2 ve 4 numaralı düğümler arasında bir kenar olması gerekmektedir. {2, 4} kümesinin CDS olabilmesi için çizgedeki 3 veya 5 numaralı düğümlerden en az birinin küme içerisine dahil edilmesi gerekmektedir. Sonuç olarak {2, 3, 4} ve {2, 4, 5} kümeleri baskın küme olduklarından ve kümeler içerindeki düğümlerin birbirlerine bağlı olmasından dolayı bağlı baskın kümelerdir.

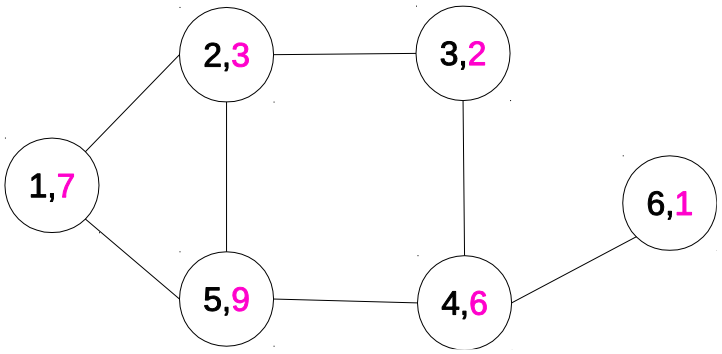
Minimum bağlı baskın kümeler (MCDS) küme büyüklüğünün minimum olduğu kümelerdir. Yukarıdaki örnekte verilen {2, 3, 4} kümesi bir CDS olmasına rağmen bir MCDS değildir. Çünkü {2, 3, 4} kümesinin büyüklüğü 3'tür ve daha küçük bir CDS bulunmaktadır. Şekil 4.1'de gösterilen çizge için MCDS = {4, 5}'tir. Fakat minimum bağlı baskın küme bulma problemi için henüz polinom zamanda bir algoritma bulunamadığından NP-Zor bir problemdir (Garey ve Johnson, 1979). Günümüze kadar geliştirilen CDS algoritmaları minimuma yaklaşan çözümler üretmektedir.

## 4.2 CDS Çeşitleri

Bağlı baskın kümeler kullanım amaçlarına göre çeşitlilik gösterebilmektedir. Bu çeşitlilik sonucunda da yeni problemler ortaya çıkmaktadır. Bu problemlerden bazıları minimum ağırlıklı CDS, minimum yönlendirme maliyetli CDS ve k-CDS'tir.

### 4.2.1 Minimum Ağırlıklı CDS

Minimum bağlı baskın küme probleminde düğümler arasında bir farklılık bulunmamaktadır ve eşdeğer sayılmaktadır. Bazı durumlarda düğümlerin birbirleriyle eşdeğer tutulması istenilmemektedir ve her bir düğüme bir ağırlık verilmektedir. Bu tarz düğümlerden oluşan çizgelere düğüm-ağırlıklı çizgeler denilmektedir. Düğüm-ağırlıklı çizgelerde CDS oluşturulduktan sonra CDS içerisindeki düğümlerin ağırlıklarının toplamının minimum olması istenilmektedir. Toplam ağırlığın minimum olduğu CDS'i oluşturma problemine Minimum Ağırlıklı CDS (MWCDS) denmektedir (Dagdeviren ve diğ. 2015).

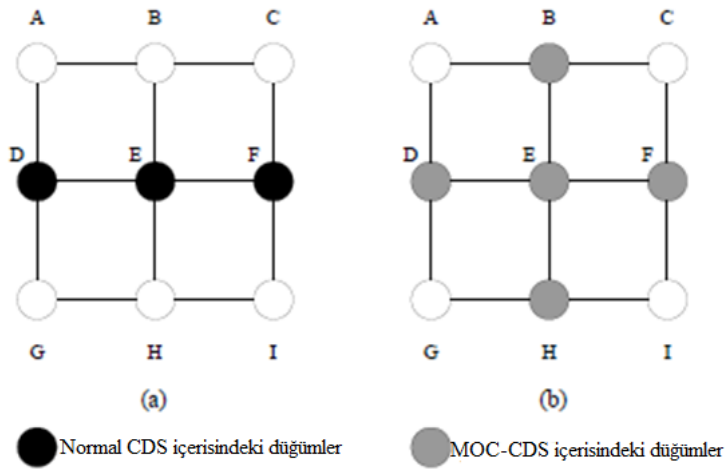



Şekil 4.2: Düğüm-ağırlıklı çizge örneği.

Şekil 4.2'de gösterilen çizge Şekil 4.1'de gösterilen çizgenin ağırlık eklenmiş halidir. Şekil 4.2'deki çizge için bulunan MCDS = {4, 5}'tir. Fakat istenilen CDS minimum ağırlıklı bir CDS ise {4, 5} kümesi istenilen sonu vermemektedir. Çünkü {4, 5} kümesinin toplam ağırlığı 9 + 16 = 15'tir ve 15 değeri bu çizge için minimum değildir. Şekil 4.2'deki çizge için minimum ağırlıklı bağlı baskın küme MWCDS = {2, 3, 4}'tür ve toplam ağırlığı 3 + 2 + 6 = 11'dir.

### 4.2.2 Minimum Yönlendirme Maliyetli CDS

Bağlı baskın kümeler düğümler arası iletişimde omurga olarak kullanılmaktadır. Bir düğümün iletişim kurmak istediği düğüm, o düğümün komşusu değil ise iletişim CDS içerisindeki düğümler kullanılarak gerçekleşir. Şekil 4.3 (a)'da örnek bir çizge ve bu çizge için bir CDS gösterilmiştir. Çizgede A düğümü C düğümüne mesaj göndermek istediği zaman, C düğümü A düğümünün komşusu olmadığı için A düğümü mesajını CDS üzerinden gönderecektir. Mesajın gideceği yol {A, D, E, F, C} yoludur.

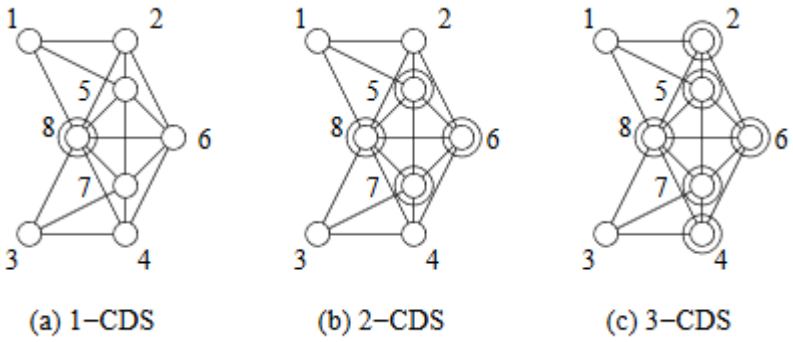



Şekil 4.3: CDS ve MOC-CDS örneği (Ding ve diğ. 2010).

Şekil 4.3 (a)'da gösterilen çizgede A'dan C'ye gönderilecek mesaj en kısa {A, B, C} yolu ile gönderilir. A düğümü C düğümüne mesaj gönderirken {A, B, C} yolu yerine {A, D, E, F, C} yolunu kullanırsa yönlendirme maliyeti artmış olacaktır. Bu gibi durumları engellemek için minimum yönlendirme maliyetli CDS (MOC-CDS)'e ihtiyaç duyulmaktadır. MOC-CDS öyle bir alt kümedir ki: çizge içerisindeki her iki düğüm arasında bulunan en kısa yolların geçtiği düğümler CDS 'in içerisindedir (Ding ve diğ. 2010). Şekil 4.3 (b)'de örnek bir MOC-CDS gösterilmiştir.

### 4.2.3 k-Baskın ve k-CDS

Bağlı baskın kümeler ağlarda hayati öneme sahiptirler ve CDS içerisindeki düğümlerden herhangi birinin bozulması durumunda ağın yaşamı da tehlikeye girebilir. CDS içerisindeki düğümler iletişim için fazla kullanıldıklarından diğer düğümlere göre enerjilerini daha hızlı tüketmektedir ve enerjileri bittiğinde çalışamamaktadır. Bu gibi durumlarda ağın hata toleransını ve esnekliğini artırmak için k-CDS'lere ihtiyaç duyulmaktadır. k-CDS'i diğer CDS'lerden ayıran iki özelliği vardır. Bunlardan birincisi oluşturulan CDS içerisindeki düğümlerin CDS içerisinde en az k komşusu vardır. İkinci özelliği ise CDS dışında kalan düğümlerin de CDS içerisinde k komşusu olmasıdır. Bu sayede CDS içerisindeki bir düğüm devre dışı kalsa bile ağ hayatını sürdürebilecektir (Dai ve diğ. 2005).




Şekil 4.4: k değerlerine göre k-CDS'lerin değişimi (Dai ve diğ. 2005).

Şekil 4.4'te k değerinin değişimine göre ağda oluşturulan k-CDS'lerinin değişimi gösterilmiştir.

## 4.3 CDS Oluşturma Algoritmaları

Kablosuz ağların popülerleşmesi ile CDS çok önem kazanmış ve birçok araştırmacı tarafından CDS oluşturma algoritmaları geliştirilmiştir. Bu algoritmaların bir kısmı merkezi bir kısmı da dağıtık algoritmalardır. Merkezi algoritmalar, ağın tüm topolojisinin tek bir bilgisayar tarafından bilindiği ve CDS oluşturma algoritmasının bu bilgisayar üzerinden yapıldığı algoritmalardır. Dağıtık algoritmalar ise ağı oluşturan düğümlerin kendi aralarında haberleşerek ağın genel topolojisini bilmeden çalıştırdıkları algoritmalardır.

CDS oluşturma algoritmaları tek fazlı ve çift fazlı olmak üzere ikiye bölünebilir. Tek fazlı CDS oluşturma algoritmaları genellikle bir veya birkaç başlangıç düğümü seçer ve ardından seçilecek düğümleri seçilen düğümlerin komşularından seçerek CDS oluşana kadar devam eder. Çift fazlı düğümler ise iki fazdan oluşur. İlk fazda çizge üzerinde DS oluşturulur. DS oluşturma işlemi bittikten sonra oluşturulan DS içerisindeki düğümleri birbirlerine bağlayan düğümler DS içerisine dahil edilerek CDS oluşturulur (Du ve Pardalos, 2005).

### 4.3.1 Steiner Ağacı Tabanlı Algoritmalar

Steiner Ağacı problemi, düğüm-ağırlıklı bir çizge içerisinde önceden belirlenen düğümler aralarında bağlantı oluşturacak düğümlerin minimum ağırlık taşıyacak şekilde seçilmesi problemidir ve NP-Zor bir problemdir (K. ve S., 1992). Steiner Ağacı problemi genellikle çift fazlı CDS oluşturma algoritmalarında kullanılmaktadır. Bu algoritmaların ilk fazında oluşturulan DS Steiner Ağacı problemi ile bağlı hale getirilerek CDS oluşturulur.

### 4.3.2 Maksimum Bağımsız Küme Tabanlı Algoritmalar

Maksimum bağımsız küme (MIS) problemi, bir çizge içerisinde birbirine komşu olmayan en fazla sayıda düğüm bulma problemidir ve NP-Tam bir problemdir (Jian, 1986). Maksimum bağımsız küme tabanlı algoritmalar dağıtık CDS oluşturma algoritmalarının büyük bir kısmını oluşturmaktadır. Bu algoritmalarda genellikle ilk fazda MIS oluşturulmaktadır ve ikinci fazda MIS içerisindeki düğümleri birbirlerine bağlayan düğümler CDS oluşturmaktadır (Mohanty ve diğ. 2017).

### 4.3.3 Budama Tabanlı Algoritmalar

Budama tabanlı algoritmalarda öncelikle bir CDS oluşturulur. CDS oluşturulduktan sonra CDS içerisinde olmaması gereken fazla düğümler CDS'ten çıkartılır. Yapılan bu işleme budama işlemi denir.

# 5. MOC-CDS ve MWOC-CDS Algoritmaları

Tezin bu bölümünde Ding ve diğ. (2010)'nin geliştirdiği minimum yönlendirme maliyetli CDS algoritması olan MOC-CDS ile konuyla ilgili olarak geliştirdiğimiz yeni minimal ağırlıklı ve yönlendirme maliyetli CDS algoritması olan MWOC-CDS algoritmaları incelenmiştir.

## 5.1 MOC-CDS

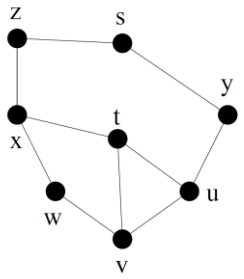
Ding ve diğ. (2010) CDS'lerin en çok yönlendirme işlemlerinde kullanıldığını ve bu amaçla bir CDS geliştirilmesi gerektiğini savunmuşlardır. Bu amaç doğrultusunda birbirleri ile haberleşmek isteyen düğümlerin yönlendirme maliyetlerini minimale indiren MOC-CDS algoritmasını geliştirmişlerdir. Algoritmada yönlendirme maliyeti olarak iki düğüm arasındaki kenar sayısı kullanılmıştır. Ding ve diğ. , birbirlerine 2-kenar uzaklıkta bulunan tüm düğümlerin arasındaki düğüm CDS içerisinde bulunursa, aralarındaki uzaklığa bağlı kalmadan herhangi düğüm ikilisinin aralarındaki en kısa yolun CDS içerisinde olacağını kanıtlamışlardır. Bu sonuçtan yola çıkarak problemi 2-kenar CDS problemine indirgemişlerdir.

2-kenar CDS (MOC-CDS) algoritmasının genel mantığı şu şekildedir: Bir düğüm daha önce birbirlerine bağlanmamış iki komşusu arasında bağlantı görevi görüyorsa o düğüm DTOR olmalıdır. CDS'in boyutunun minimum olması için bir düğüm tarafından bağlanılan düğümlerin, başka bir düğüm tarafından bağlanmalarına gerek yoktur. Bu yüzden algoritma ilk olarak, fazla bağlantı kurabilen düğümleri diğerlerinden daha önce DTOR olarak seçmektedir. Her turda birbirine bağlanacak düğüm ikilileri azalmaktadır. Birbirlerine bağlanacak düğüm ikilisi kalmadığında algoritma sonlanmakta ve MOC-CDS oluşmaktadır.

### 5.1.1 MOC-CDS Algoritması

Algoritma disk çizgeler düşünülerek tasarlanmıştır. Dolayısıyla düğümler mesaj aldıkları düğümlere mesaj gönderemeyebilirler. Çünkü her düğümün menzili farklıdır. Algoritmada düğümler 2-kenar komşuluk bilgisini kullanmaktadır. Bu yüzden düğümler algoritmaya başlamadan önce 2-kenar komşuluk bilgilerini toplamak zorundadır.

2-kenar komşuluk bilgisini toplamak için düğümler belirli aralıklarla belirli sayıda ***Hello*** mesajları göndermektedir. ***Hello*** mesajı $N_{in}(m)$ ve $N_{out}(m)$ kümelerini içermektedir. $N_{in}(m)$ kümesi ***m*** düğümünün mesajını alabildiği düğümler kümesidir. $N_{out}(m)$ kümesi ise ***m*** düğümünün mesajını alabilen düğümler kümesidir. Başlangıçta kümeler boştur ve düğümlerin diğer düğümler hakkında herhangi bir bilgisi yoktur. Bir düğüm ***Hello*** mesajını aldığında mesajı gönderen düğümü $N_{in}(m)$ kümesine eklemektedir. Alınan ***Hello*** mesajının içerisindeki $N_{in}(w)$ kümesinin içerisinde kendini görürse ***w*** düğümünü $N_{out}(m)$ kümesine eklemektedir. ***Hello*** mesajları birkaç kez tekrarlandıktan sonra düğümler mesaj alıp ve gönderebildikleri düğümleri komşuları, komşularının mesaj alıp-gönderebildikleri ve kendi komşusu olmayan düğümleri de 2-kenar komşusu olarak belirlemektedir.




Şekil 5.1: MOC-CDS gösterimi için bir çizge (Ding ve diğ. 2010).

2-kenar komşulukları belirlendikten sonra algoritma çalışabilir hale gelmektedir. İlk olarak düğümler kendi ***P(m)*** kümesini ve ***f(m)*** değişkenini hesaplamaktadır. ***P(m)*** kümesi ***m*** düğümünün aralarında bağlantı kurduğu, kendi aralarında komşu olmayan düğüm ikililerinden oluşur. ***f(m)*** ise ***P(m)*** kümesinin eleman sayısıdır. Örneğin Şekil 5.1'de ***v*** düğümü için ***P(v)*** kümesi, ***v*** düğümünün aralarında bağlantı kurduğu ***(w, t)*** ve ***(w, u)*** ikili düğümlerinden oluşmaktadır. ***(t, u)*** ikilisi ***P(v)*** içerisine dahil edilmez çünkü ***t*** ve ***u*** düğümleri zaten birbirlerine

komşudur ve bağlantıya ihtiyaç duymazlar. Sonuç olarak ***P(v) = {(w, t), (w, u)}*** ve ***f(v) = P(v)*** olduğundan ***f(v) = 2*** elde edilmektedir.

***P(m)*** kümesi ve ***f(m)*** değişkeni hesaplandıktan sonra algoritmada bayrak yarışı uygulanmaktadır. Her düğüm komşularından ***f*** değerlerini alır ve en büyük ***f*** değerine sahip olan düğüme bayrak gönderir. Tüm komşularından bayrak toplayan düğüm kendini DTOR ilan etmektedir.

**Algoritma 1:** m düğümü için MOC-CDS algoritması

**Adım 1.** *P(m)* kümesini ve *f(m)* değişkenini hesapla ve *f(m)* değişkenini tüm komşularına gönder.

**Adım 2.** kendisi ve komşuları içerisindeki maksimum *f* değerini bul. Maksimum *f* değerine sahip birden fazla düğüm varsa ID'si büyük olan düğümü seç ve o düğüme bayrak gönder.

**Adım 3.** Tüm komşulardan bayrak alındıysa kendini DTOR olarak işaretle. *P(m)* kümesini tüm komşularına gönder ve *P(m)* kümesini boşalt.

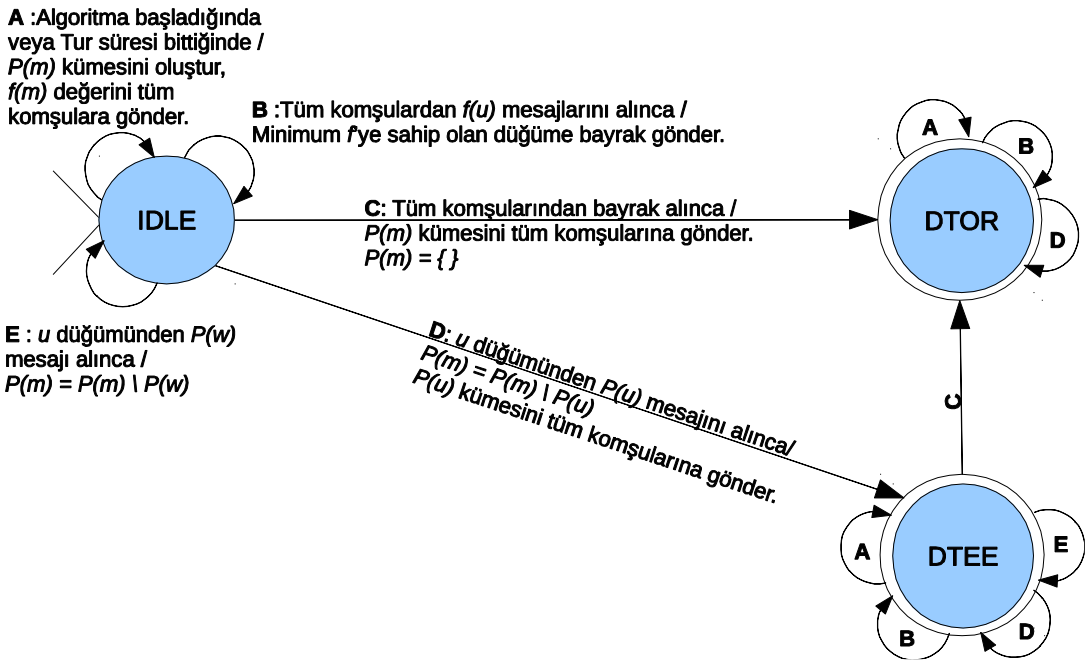
**Adım 4.** Herhangi bir *u* komşusundan *P(u)* kümesi alındıysa *P(u)* kümesini tüm komşularına gönder ve *P(m)* kümesinden *P(u)* kümesindeki düğüm ikililerini çıkart.

**Adım 5.** Herhangi bir *w* komşusundan *P(u)* kümesi alındıysa *P(m)* kümesinden *P(u)* kümesindeki düğüm ikililerini çıkart.

Herhangi bir v düğümü için MOC-CDS algoritmasının bir turu Algoritma 1'de verilmiştir. Algoritmanın 1. adımında m düğümü, aralarında bağlantı görevi gördüğü düğüm ikililerini tespit etmekte ve ***P(m)*** kümesini ve ***f(m)*** değerini hesaplamaktadır. Ardından hesaplanan ***f(m)*** değerini tüm komşularına göndermektedir. 2. adımda tüm komşularından ***f(u)*** değerlerini biriktiren m düğümü,

kendisi de dahil olmak üzere komşuları içerisinden maksimum ***f*** değerine sahip düğümü bulmaktadır. Maksimum ***f*** değerine sahip birden fazla düğüm varsa ID'si büyük olan düğüm seçilmektedir. Seçilen düğüm m düğümünün kendisi değil ise seçilen komşuya bayrak mesajı gönderilmektedir. 3. adımda m düğümü tüm komşularından bayrakları aldıysa kendisini DTOR yapmaktadır. Bu noktada ***P(m)*** kümesindeki ikililer birbirlerine bağlanmış olup diğer düğümlerin tekrardan bağlamasına gerek yoktur. Diğer düğümlerin ***m*** düğümünün bağladığı düğüm ikililerini kendi ***P*** kümelerinden çıkartmaları için ***m*** düğümü ***P(m)*** kümesini 2-kenar uzaklıktaki düğümlere göndermesi gerekmektedir. Çünkü ***m*** düğümünün komşularını yalnızca komşularının komşuları bağlayabilir. Ardından ***m*** düğümü ***P(m)*** kümesini tekrar DTOR olarak seçilmemek için boşaltır. 4. adımda ***u*** düğümünden ***P(u)*** kümesini alan ***m*** düğümü ***P(u)*** kümesini kendi komşularına iletir. 5. adımda ***w*** düğümünden ***P(u)*** kümesini alan ***m*** düğümü ***P(u)*** kümesindeki ikilileri ***P(m)*** kümesinden çıkarmaktadır. Örneğin: Şekil 5.1 'de ***f(x) = 3*** değeri komşuları içerisinde maksimum değer olduğu için tüm komşularından bayrakları toplayan ***x*** düğümü DTOR olarak seçildikten sonra ***P(x) = {(z, t), (z, w), (t, w)}*** kümesini komşuları ***z, t*** ve ***w*** düğümlerine gönderir. ***P(x)*** kümesini alan ***z, t*** ve ***w*** düğümleri ***P(x)*** kümesini kendi komşuları olan ***s, u*** ve ***v*** düğümlerine gönderir. ***P(x)*** kümesini alan ***v*** düğümü ***P(v)*** kümesinden ***P(x)*** kümesinde olan ***(t, w)*** ikilisini çıkartır. Daha sonra ***P(x) = { } ve P(v) = {(u, w)}*** olur.

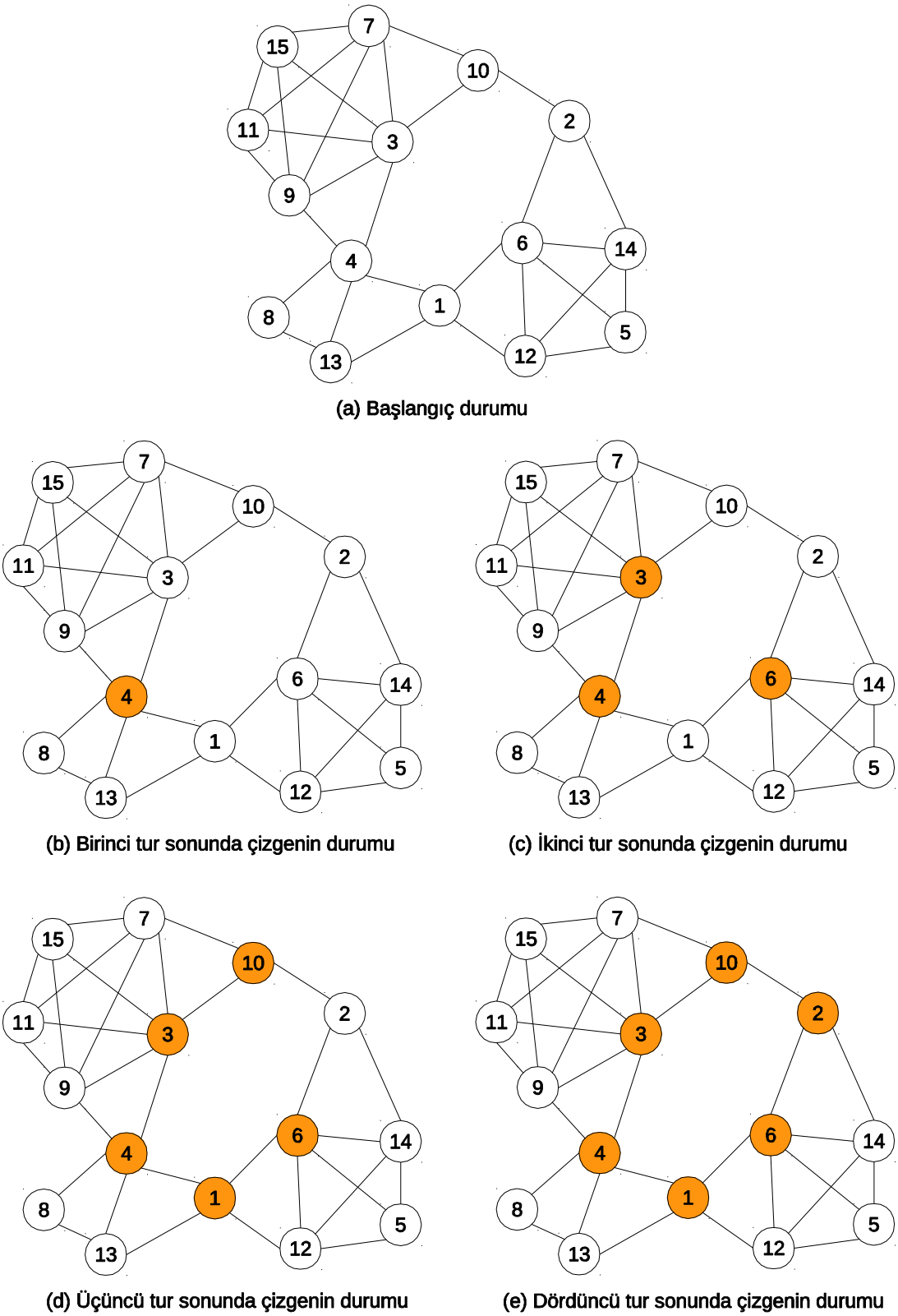
Bu tur çizgedeki her ***m*** düğümü için ***f(m) = 0*** olana kadar tekrarlanmakta ve sonuç olarak MOC-CDS oluşmaktadır. Şekil 5.2'de ***m*** düğümü için MOC-CDS algoritmasının sonlu durum diyagramı verilmiştir.




Şekil 5.2: MOC-CDS algoritmasının sonlu durum diyagramı.

### 5.1.2 MOC-CDS Algoritmasının Örnek Üzerinde Gösterimi

Şekil 5.3'de MOC-CDS algoritmasının örnek bir çizge üzerinde işleyişi verilmiştir. Şekil 5.3 (a)'da 15 düğümden oluşan yönsüz ve ağırlıksız bir çizgenin başlangıç durumu verilmiştir.




Şekil 5.3: MOC-CDS algoritmasının örnek işleyişi.

Algoritmanın birinci turunda tüm düğümler ***P*** kümelerini ve ***f*** değerlerini hesaplamaktadır. Hesaplanan ***P*** kümelerinden ve ***f*** değerlerinden bazıları aşağıdaki gibidir:

***P(1)*** = {(4, 6), (4, 12), (6, 13), (12, 13)}, ***f(1)*** = 4

***P(3)*** = {(4, 7), (4, 10), (4, 11), (4,15), (9, 10), (10, 11), (10, 15)}, ***f(3)*** = 7

***P(4)*** = {(1, 3), (1, 8), (1, 9), (3, 8), (3, 13), (8, 9), (9, 13)}, ***f(4)*** = 7

***P(6)*** = {(1, 2), (1, 5), (1, 14), (2, 5), (2, 12)}, ***f(6)*** = 5

***P(13)*** = {(1, 8)} , ***f(13)*** = 1

Bu durumda ***1, 8, 9*** ve ***13*** numaralı düğümler daha büyük ***f*** değerine sahip olan ***4*** numaralı düğüme; ***7, 10, 11*** ve ***15*** numaralı düğümler ***3*** numaralı düğüme; ***2, 5, 12*** ve ***14*** numaralı düğümler ***6*** numaralı düğüme; ***3*** numaralı düğüm aynı ***f*** değerine sahip olmasına rağmen ID'si daha büyük olan ***4*** numaralı düğüme bayraklarını göndermektedirler. ***4*** ve ***6*** numaralı düğüm komşuları arasında en yüksek ***f*** değerlerine sahip oldukları için bayrak göndermemektedirler. ***6*** numaralı düğüm ***1*** numaralı düğümden bayrak alamadığı için DTOR olamamaktadır. Tüm komşularından bayrakları toplayan ***4*** numaralı düğüm DTOR olmaktadır ve ***P(4)*** kümesini komşularına gönderdikten sonra ***P(4)*** kümesini boşaltmaktadır. ***P(4)*** kümesini alan ***1, 3, 8, 9*** ve ***13*** numaralı düğümler ***P(4)*** kümesindeki ikilileri kendi ***P*** kümelerinden çıkartmaktadır ve ***P(4)*** kümesini komşularına göndermektedirler. ***6, 7, 10, 11, 12*** ve ***15*** numaralı düğümler ***P(4)*** kümesini aldıktan sonra kendi ***P*** kümelerini güncellemektedir ve 1. tur sonlanmaktadır. 1. turun sonunda ***P*** kümesi değişen düğümler ***4*** ve ***13*** numaralı düğümlerdir ve ***P(4) = { }, P(13) = { }*** olmaktadır.

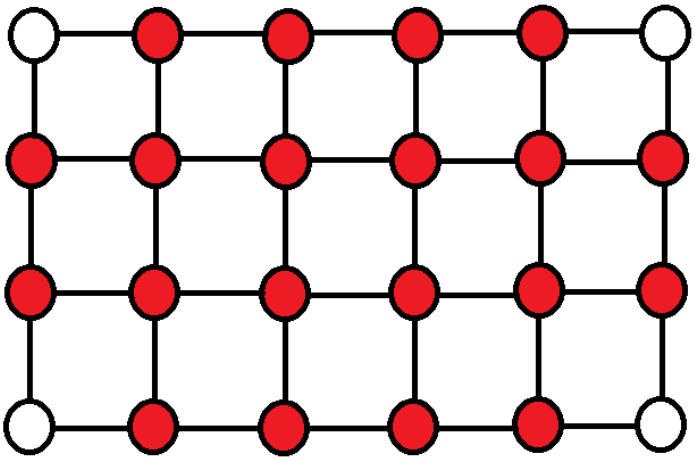
İkinci turda da tüm düğümler ***f*** değerlerini komşularına göndermektedir. ***F*** değerlerini toplayan düğümler komşuları içerisinde en büyük ***f*** değerine sahip olan düğümler ***3*** ve ***6*** numaralı düğümlerdir ve tüm bayrakları toplayarak DTOR olmaktadırlar. DTOR olan düğümler kendi ***P*** kümelerini kendi komşularına göndermektedirler ve ***P*** kümelerini boşaltmaktadırlar. ***P(3)*** veya ***P(6)*** kümesini alan düğümler kendi ***P*** kümelerini güncelledikten sonra aldıkları ***P(3)*** veya ***P(6)*** kümesini kendi komşularına iletirler. Bu mesajları alan düğümler ***P*** kümelerini

güncellemektedirler. İkinci tur sonunda oluşan durum Şekil 5.3 (c)'de gösterilmektedir.

Sonraki turlarda da aynı şekilde çalışan algoritma 3. turda ***1*** ve ***10*** numaralı düğümleri, son turda ***2*** numaralı düğümü DTOR olarak seçmektedir. Son turdan sonra çizge Şekil 5.3 (e)'deki hali almaktadır. Mavi renk gösterilen ***1,2,3,4,6*** ve ***10*** numaralı düğümler DTOR düğümlerdir.

## 5.2 MWOC-CDS

Telsiz duyarga ağlarında düğümler enerjilerini verimli kullanabilmek için MOC-CDS'i kullanabilirler. Fakat zaman içinde bazı düğümler daha az, bazı düğümler daha fazla enerji harcayabilirler ve tüm düğümlerin enerjileri eşit seviyede olmayabilir. Bunun gibi enerjinin dengesiz dağıldığı ağlarda MOC-CDS iyi bir çözüm olmayabilir. Örneğin çok düşük enerji seviyesine sahip bir düğümün kalan enerjisi ağ üzerinde omurga görevi görmek için kullanılırsa asıl amacı olan ortamdaki fiziksel veriyi ölçüp işlenmesi ve baz bilgisayara gönderim işlemleri için yeterli enerjisi kalmayabilir. Bu yüzden TDA için kullanılan ağ modelinde özdeş düğümler yerine ağırlıklı düğümler kullanılmalıdır.



Şekil 5.4: Örnek bir MOC-CDS.

Genel olarak CDS'lerin amacı ağdaki tüm düğümler yerine bazı düğümleri omurga olarak kullanmaktır. MOC-CDS algoritması ile Şekil 5.4'teki gibi simetrik çizgelerde neredeyse düğümlerin tamamı DTOR olarak seçilmekte ve omurga görevi

görmektedir. Bu durumlarla başa çıkmak için hem düğümlerin ağırlığı göz önünde bulundurulmalı hem de düğümler arası minimum yönlendirme maliyetinden taviz verilmelidir. Tez kapsamında bu parametreleri esas alan yeni bir Minimal Ağırlık ve Yönlendirme Maliyetli Bağlı Baskın Küme (MWOC-CDS) algoritması önerilmiştir.

### 5.2.1 MWOC-CDS Algoritması

MWOC-CDS algoritmasında başlangıçta tüm düğümler IDLE durumdadır ve düğümlerin kendi enerjileri ile ters orantılı olan ağırlıkları ($w_m$) ile komşularını ($\Gamma_m$) bilmektedir. $w_m$, $m$ düğümünün ağırlığını, $\Gamma_m$ ise $m$ düğümünün komşularını temsil etmektedir. Düğümler ilk olarak komşuluk bilgilerini tüm komşularına göndermektedir. Düğümler tüm komşularının komşuluk bilgilerini aldıklarında 2-kenar komşuluk bilgilerini toplamış olmaktadır.

Düğümler 2-kenar komşuluk bilgilerini topladıktan sonra $P_m$ kümesini ve $f_m$ değerini hesaplamaktadırlar. $P_m$ kümesi $m$ düğümünün komşusu olan ve kendi aralarında komşu olmayan IDLE düğüm ikililerinden oluşmaktadır. $f_m$ değeri ise $P_m$ kümesinin eleman sayısıdır. Örneğin Şekil 5.1'deki çizgedeki $t$ düğümü için $P_t$ kümesi $\{(x, v), (x, u)\}$ şeklindedir. Çünkü başlangıçta tüm düğümler IDLE durumdadır ve $t$ düğümünün komşusu olup birbiri aralarında komşu olmayan ikililer $(x, v)$ ve $(x, u)$ ikilileridir. $f_t$ değeri ise $P_t$ kümesinin elaman sayısı olduğundan $|P_t| = 2$'dir.

MWOC-CDS algoritmasında düğümler seçilirken ağırlık oranları kullanılmaktadır. Algoritmada $WR$ ve $WI$ olmak üzere iki ağırlık oranı bulunmaktadır. $WR_m$ ağırlık oranı $m$ düğümün ağırlığının ($w_m$), $m$ düğümünün $f_m$ değerine oranıdır. $WI_m$ ağırlık oranı ise $m$ düğümün ağırlığının ($w_m$), $m$ düğümünün IDLE komşu sayısı ($I_m$) ile toplam komşu sayısının ($|\Gamma_m|$) çarpımına oranıdır. Örneğin Şekil 5.1'deki $t$ düğümü için $w_t = 5$ varsayıldığında, $WR_t = 5/2 = 2.5$ şeklindedir. $t$ düğümünün tüm komşuları IDLE olduğu için $I_t = 3$ ve $|\Gamma_t| = 3$'tür. Dolayısıyla $WI_t = 5 / ( 3 \times 3) = 0.55$ şeklindedir.

$WR_m$ ve $WI_m$ değerleri hesaplandıktan sonra algoritmada MOC-CDS'teki gibi bayrak yarışı uygulanmaktadır. Tüm DTOR olmayan komşularından $WR$ ve $WI$

değerlerini toplayan düğüm $\boldsymbol{WR_{min}}$ değerine sahip olan düğüme bayrak göndermektedir. $\boldsymbol{WR_{min}}$ değeri sonsuz ise bayrak minimum $\boldsymbol{WI_{min}}$ değerine sahip olan düğüme gönderilir. $\boldsymbol{WR_{min}}$ veya $\boldsymbol{WI_{min}}$ değerlerinde eşitlik ile karşılaşılırsa ID'si küçük olan seçilerek bayrak gönderilmektedir. Bir düğüm DTOR olmayan tüm komşularından bayrakları toplarsa kendini DTOR ilan etmektedir.

---

**Algoritma 2:** *m* düğümü için MWOC-CDS algoritması

---

**Tur başladığında** $P_m$ kümesini oluştur. $f_m$ değerini $WR_m$ ve $WI_m$ ağırlık oranlarını hesapla ve $WEIGHTRATIO_m$ mesajını DTOR olmayan tüm komşularına gönder.

**Tüm *WEIGHTRATIO* mesajlarını aldığında** $WR_{min}$'i hesapla. $WR_{min}$ sonsuz ise $WI_{min}$'i hesapla. $WI_{min}$ sonsuz değilse $WI_{min}$ ağırlık oranına sahip olan komşuya *FLAG* mesajını gönder.

**DTOR olmayan tüm komşulardan *FLAG* mesajını aldığında** kendi durumunu DTOR yap. $DTOR_m$ mesajını tüm komşularına gönder.

***i* düğümünden $\boldsymbol{DTOR_i}$ mesajı aldığında** *i* düğümünün durumunu *DTOR* yap. İçinde *i* düğümü bulunan ikilileri $P_m$ kümesinden çıkar. Durum IDLE ise kendi durumunu DTEE yap. $DTEE_m$ mesajını tüm komşularına gönder.

***i* düğümünden $\boldsymbol{DTEE_i}$ mesajı aldığında** *i* düğümünün durumunu DTEE yap. İçinde *i* düğümü bulunan ikilileri $P_m$ kümesinden çıkar.

---

Algoritma 2'de 2-kenar komşuluk bilgilerine sahip ***m*** düğümü için MWOC-CDS algoritması verilmiştir. Algoritmaya göre ilk adımda $\boldsymbol{P_m}$ kümesi oluşturulmakta ve $\boldsymbol{f_m}$ değeri hesaplanmaktadır. $\boldsymbol{P_m}$ kümesi ve $\boldsymbol{f_m}$ değeri hesaplandıktan sonra $\boldsymbol{WR_m}$ ve $\boldsymbol{WI_m}$ ağırlık oranları hesaplanmaktadır. Hesaplanan $\boldsymbol{WR_m}$ ve $\boldsymbol{WI_m}$ ağırlık oranları DTOR olmaya tüm komşulara $\boldsymbol{WEIGHTRATIO_m}$ mesajı olarak gönderilmektedir.

DTOR olmayan tüm komşularından ***WEIGHTRATIO*** mesajlarını alan düğüm $WR_{min}$ ağırlık oranını hesaplamaktadır. $WR_{min}$ ağırlık oranı sonsuz ise $WI_{min}$ ağırlık oranını hesaplamaktadır. $WR_{min}$ veya $WI_{min}$ değerlerinde bir eşitlik durumunda ID'si küçük olan seçilmektedir. $WR_{min}$ ve $WI_{min}$ ağırlık oranları sonsuz ise bayrak gönderilmemektedir aksi takdirde $WR_{min}$ veya $WI_{min}$ ağırlık oranına sahip olan komşu düğüme ***FLAG*** mesajı gönderilmektedir.

DTOR olmayan tüm komşularından ***FLAG*** mesajını alan düğüm kendisini DTOR yapmaktadır ve tüm komşularına DTOR olduğunu bildirmek için $DTOR_m$ mesajını göndermektedir.

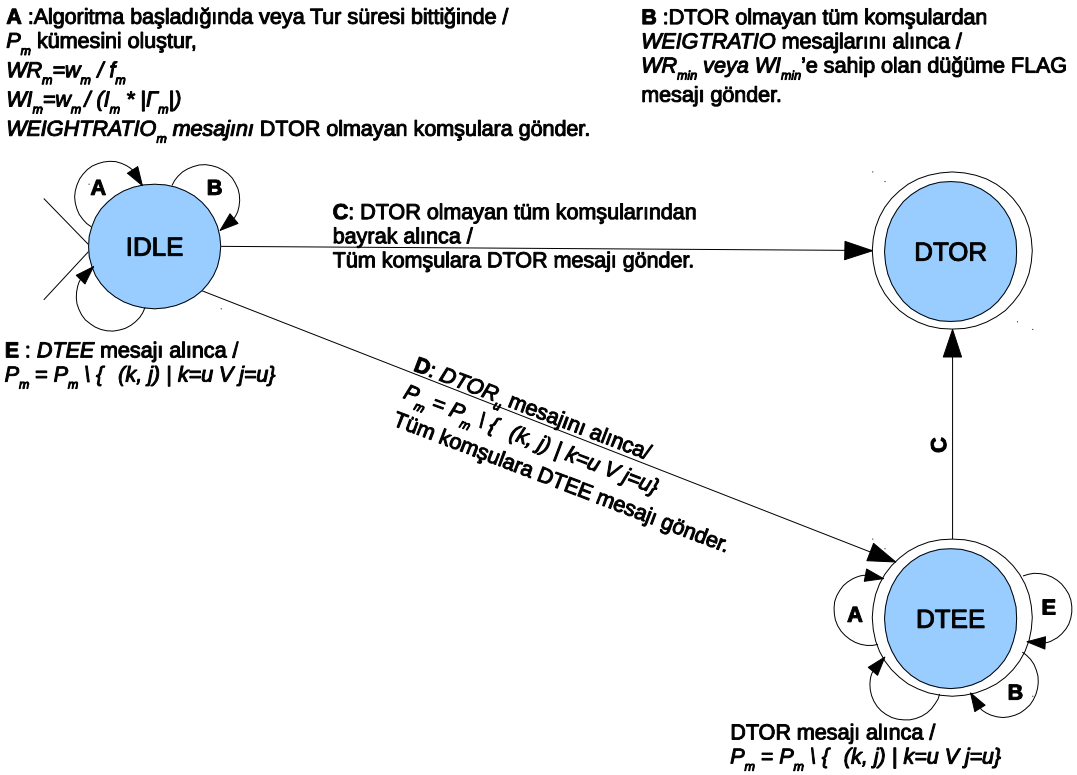
***m*** düğümü ***i*** düğümünden $DTOR_i$ mesajı alırsa ***i*** düğümünü DTOR olarak işaretlemektedir. İçinde ***i*** düğümü bulunan tüm düğüm ikililerini $P_m$ kümesinden çıkarmaktadır. Çünkü ***i*** düğümü artık IDLE değildir ve $P_m$ kümesi sadece IDLE ikililerini içinde bulundurmaktadır. ***m*** düğümünün durumu IDLE ise kendi kendini DTEE yapmaktadır ve tüm komşularına DTEE olduğunu bildirmek için $DTEE_m$ mesajını göndermektedir.

***i*** düğümünden $DTEE_i$ mesajını alan ***m*** düğümü, ***i*** düğümünü DTEE olarak işaretlemektedir. ***i*** düğümü artık IDLE olmadığı için içinde ***i*** düğümü bulunan tüm düğüm ikililerini $P_m$ kümesinden çıkarmaktadır.

Bu döngü çizgedeki her ***m*** değeri için $WR_m$ ve $WI_m$ ağırlık oranları sonsuz olana kadar tekrarlanmaktadır ve sonuç olarak bağlı olmayan baskın küme oluşturmaktadır. Algoritmanın sonunda bağlı olmayan DTOR düğümleri birbirine bağlamak için Dagdeviren ve diğ. (2015)'nin geliştirdiği Steiner Ağacı algoritması olan ASYNTREE algoritması çalıştırılmaktadır ve MWOC-CDS oluşmaktadır.

Şekil 5.5'te ***m*** düğümü için MWOC-CDS algoritmasının sonlu durum diyagramı verilmiştir. MWOC-CDS algoritmasının MOC-CDS algoritmasına göre birçok farkı bulunmaktadır. MOC-CDS algoritmasında ***f*** değerlerine göre DTOR seçimi yapılırken MWOC-CDS algoritmasında ***WR*** ve ***WI*** ağırlık oranlarına göre seçim yapılmaktadır. MOC-CDS algoritmasında ***P*** kümesi güncellenirken, DTOR olan düğümün ***P*** kümesindeki ikilileri çıkarılmaktadır. MWOC-CDS algoritmasında ise ***P*** kümesinden, içerisinde durumu DTOR veya DTEE düğüm olan tüm ikililer

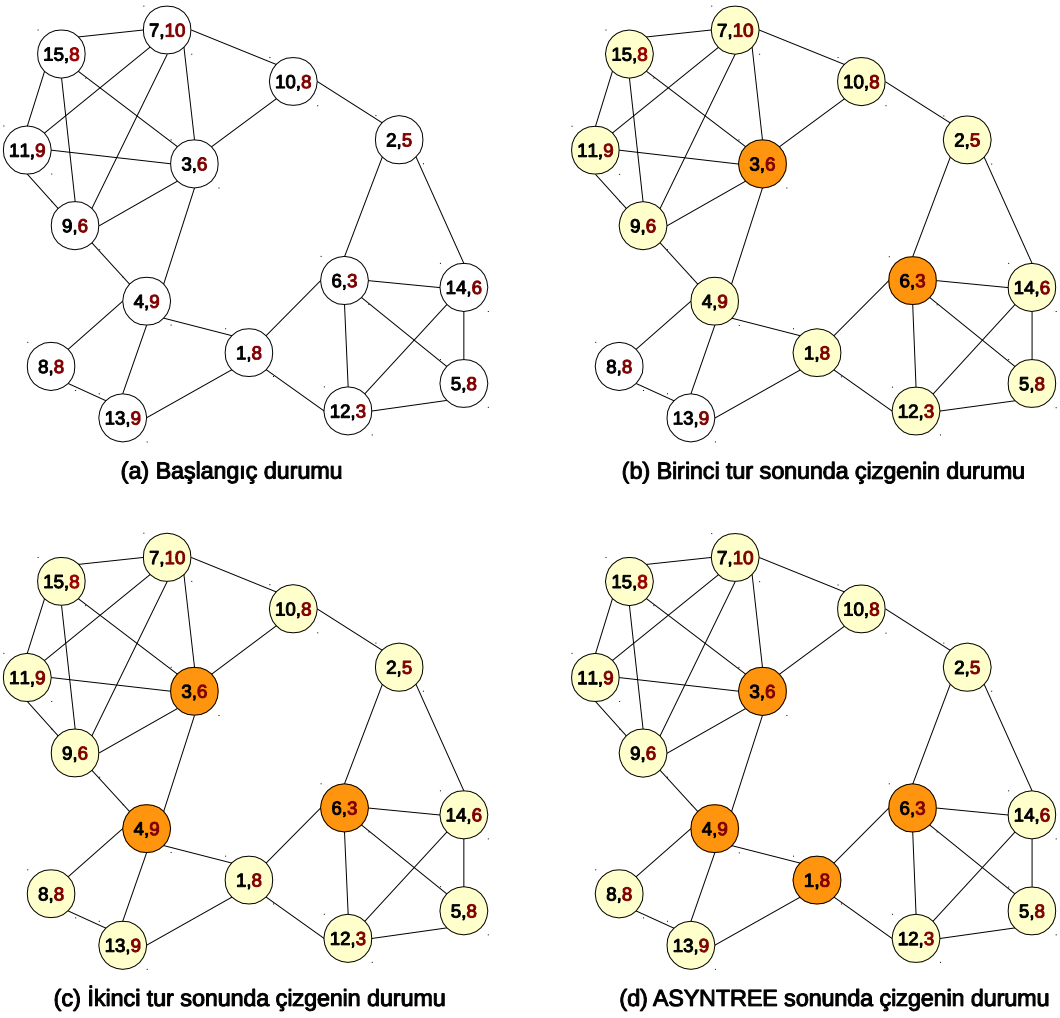
çıkarılmaktadır. MOC-CDS algoritmasında durumu DTOR olan düğüm algoritmada aktif rol almaya devam etmektedir. MWOC-CDS algoritmasında ise durumu DTOR olan düğümler algoritmada aktif rol almamaktadırlar.




Şekil 5.5: MWOC-CDS algoritmasının örnek işleyişi.

### 5.2.2 MWOC-CDS Algoritmasının Örnek Üzerinde Gösterimi

Şekil 5.6'da MWOC-CDS algoritmasının, Şekil 5.3'te verilen çizgenin ağırlıklandırılmış halinin üzerinde işleyişi verilmiştir.




Şekil 5.6: MWOC-CDS algoritmasının örnek işleyişi.

Algoritmanın başlangıcında tüm düğümler ***P*** kümelerini ***f*** değerlerini ***WR*** ve ***WI*** ağırlık oranlarını hesaplamaktadır. Hesaplanan değerlerden bazıları aşağıdaki gibidir:

$P_1$ = {(4, 6), (4, 12), (6, 13), (12, 13)} , $f_1$ = 4, $WR_1$ = 8 / 4 = 2, $WI_1$ = 8 / (4 * 4) = 0.5

$P_3$ = {(4, 7), (4, 10), (4, 11), (4,15), (9, 10), (10, 11), (10, 15)} , $f_3$ = 7, $WR_3$ = 6 / 7 = 0.86, $WI_3$ = 6 / (6 * 6) = 0.16

$P_4$ = {(1, 3), (1, 8), (1, 9), (3, 8), (3, 13), (8, 9), (9, 13)} , $f_4$ = 7, $WR_4$ = 9 / 7 = 1.28, $WI_4$ = 9 / (5 * 5) = 0.36

$P_6$ = {(1, 2), (1, 5), (1, 14), (2, 5), (2, 12)} , $f_6$ = 5, $WR_6$ = 3 / 5 = 0.6, $WI_6$ = 3 / (5 * 5) = 0.12

$P_{13}$ = {(1, 8)} , $f_{13}$ = 1, $WR_{13}$ = 9 / 1 = 9, $WI_{13}$ = 3 / (3 * 3) = 0.33

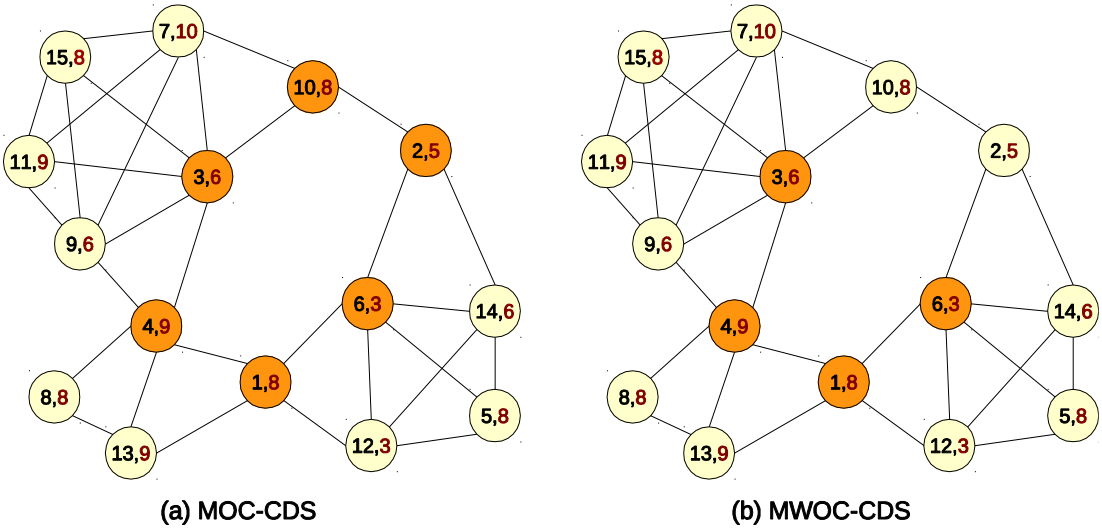
Düğümler ***WR*** ve ***WI*** değerlerini hesapladıktan sonra ***WEIGHTRATIO*** mesajını DTOR olmayan tüm komşularına göndermektedir. DTOR olmayan tüm komşularından ***WEIGHTRATIO*** mesajlarını alan düğümler $WR_{min}$ ağırlık oranını bulmakta ve $WR_{min}$ ağırlık oranına sahip olan komşusuna ***FLAG*** mesajını gönderir. ***1, 2, 5, 12*** ve ***14*** numaralı düğümler ***6*** numaralı düğüme, ***8*** ve ***13*** numaralı düğümler ***4*** numaralı düğüme, ***4, 7, 9, 10, 11*** ve ***15*** numaralı düğümler ***3*** numaralı düğüme bayrak mesajlarını göndermektedir. DTOR olmayan tüm komşularından ***FLAG*** mesajlarını alan ***3*** ve ***6*** numaralı düğümler, tüm komşularına ***DTOR*** mesajını göndermektedir. $DTOR_3$ mesajını alan ***4, 7, 9, 10, 11*** ve ***15*** numaralı düğümler içinde ***3*** numaralı düğüm olan tüm düğüm ikililerini ve $DTOR_6$ mesajını alan ***1, 2, 5, 12*** ve ***14*** numaralı düğümler içinde ***6*** numaralı düğüm olan tüm düğüm ikililerini ***P*** kümelerinden çıkartmaktadır ve IDLE durumunda oldukları için kendilerini DTEE yaptıktan sonra tüm komşularına ***DTEE*** mesajlarını göndermektedir. ***1*** ve ***4*** numaralı komşularından ***DTEE*** mesajı alan ***8*** ve ***13*** numaralı düğüm içerisinde ***1*** veya ***4*** numaralı düğümlerin bulunduğu tüm düğüm ikililerini ***P*** kümelerinden çıkarmaktadır.

1. turun sonunda tüm düğümlerin ***P*** kümeleri boş küme olmaktadır. Bu yüzden tüm düğümlerin ***WR*** ağırlık oranları sonsuz olmaktadır. ***WR*** ağırlık oranları sonsuz olduğu için 2. turda ***WI*** ağırlık oranına bakılarak DTOR seçilecektir. 2. turun başlangıcında sonsuz olmayan ***WI*** ağırlık oranları aşağıdaki gibidir:

$WI_1 = 8/(1*4)=2$, $WI_4 = 9/(2*5)=0.9$, $WR_8 = 8/(1*2)=4$, $WR_{13} = 9/(1*3)=3$

2. turda ***WEIGHTRATIO*** mesajları gönderildikten sonra ***1, 8, 9*** ve ***13*** numaralı düğümler ***FLAG*** mesajlarını ***4*** numaralı düğüme göndermektedir. DTOR olmayan tüm komşularından ***FLAG*** mesajı alan ***4*** numaralı düğüm ***DTOR*** mesajını ***1, 8, 9*** ve ***13*** numaralı düğümlere göndermektedir. ***DTOR*** mesajını alan ***8*** ve ***13*** numaralı düğümler kendilerini DTEE yapmaktadır. Çizge içerisinde IDLE düğüm kalmadığından tüm ***WR*** ve ***WI*** ağırlık oranları sonsuz olmaktadır ve bağlı olmayan baskın küme oluşmuştur. Bağlı olmayan baskın kümeyi bağlı hale getirebilmek için ***ASYNTREE*** algoritması çalıştırılarak ***1*** numaralı düğüm de DTOR seçilmektedir ve MWOC-CDS oluşturulmaktadır. Çizgenin son durumu Şekil 5.6 (d)'de verilmiştir. Mavi olan ***1,3,4*** ve ***6*** numaralı düğümler DTOR, sarı olan düğümler ise DTEE durumundadır.

MOC-CDS ve MWOC-CDS algoritmalarının örnek anlatımları aynı çizge üzerinde yapılmıştır ve sonuçları Şekil 5.7'de yan yana gösterilmiştir. Alınan sonuçlara göre MOC-CDS ve WMOC-CDS algoritmaları karşılaştırıldığında bağlı baskın kümeler içerisindeki düğüm sayısı sırasıyla 6 ve 4, oluşan kümelerin ağırlıkları toplamı ise sırasıyla 39 ve 26 olmaktadır. Baskın kümelerin bağladığı ikili düğüm sayısı MOC-CDS için 27, MWOC-CDS için 23 olmaktadır.




Şekil 5.7: MOC-CDS ve MWOC-CDS örnek sonuçları.

## 5.3 MOC-CDS ve MWOC-CDS Algoritmalarının Benzetim Sonuçları

MOC-CDS ve MWOC-CDS algoritmaları Contiki (Dunkels ve diğ. 2004) işletim sistemi üzerinde çalışacak şekilde kodlanmıştır. Benzetim ortamı olarak ise COOJA (Österlind ve diğ. 2006) kullanılmıştır. Algoritmaların benzetimi yapılırken ağ topolojilerini temsil eden, 10, 20, 30 ve 50 düğümden oluşan çizgeler kullanılmıştır. Çizgeler tamamen rastgele oluşturulmuş birim disk çizgeleridir.

Tablo 5.1'de MOC-CDS algoritmasının benzetim sonuçları, Tablo 5.2'de ise MWOC-CDS algoritmasının benzetim sonuçları verilmiştir. Tablodaki Toplam DTOR Sayısı, algoritma çalıştıktan sonra oluşan CDS içerisindeki düğüm sayısıdır. Toplam CDS Ağırlığı, oluşan CDS içerisindeki düğümlerin ağırlıklarının toplamıdır. Toplam 2-kenar Sayısı, oluşan CDS içerisindeki düğümlerin bağladığı düğüm ikilisi sayısının toplamıdır. Toplam En Kısa Yol Kenar Sayısı, çizgedeki düğümlerin diğer

düğümlere erişebilmesi için, CDS içerisindeki düğümleri kullanarak buldukları en kısa yol içerisindeki kenar sayılarının toplamıdır. Toplam En Kısa Yol Ağırlığı ise buldukları en kısa yollar üzerindeki düğümlerin ağırlıklarının toplamıdır.

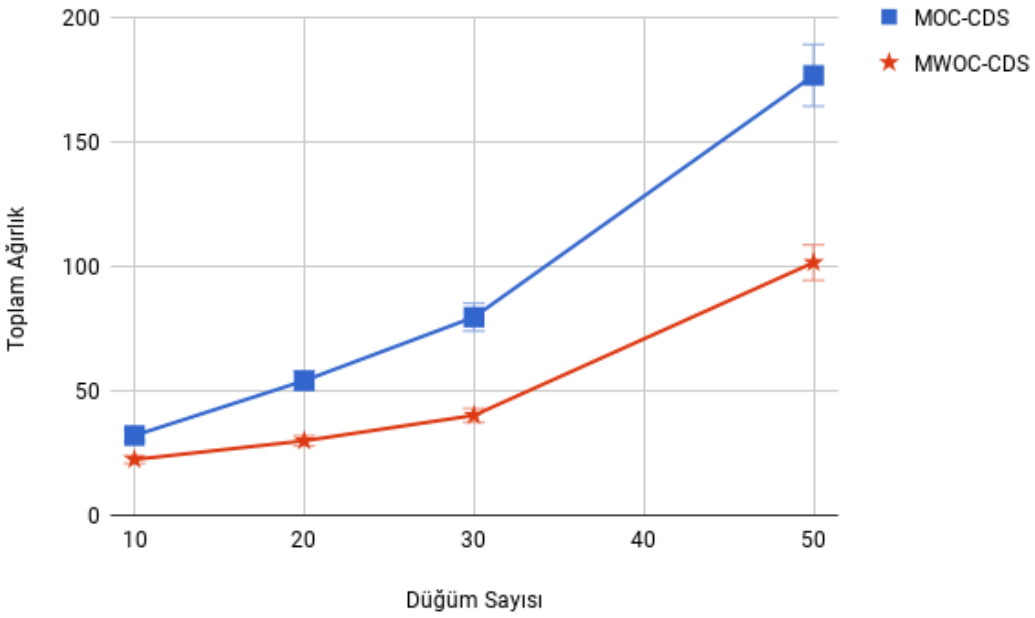
Tablo 5.1: MOC-CDS algoritması benzetim sonuçları.

| Çizgedeki Düğüm Sayısı | Toplam DTOR Sayısı | Toplam CDS Ağırlığı | Toplam 2-kenar Sayısı | Toplam En Kısa Yol Kenar Sayısı | Toplam En Kısa Yol Ağırlığı |
|---|---|---|---|---|---|
| 10 | 5,7 | 32 | 16 | 189,4 | 524,8 |
| 20 | 9,7 | 54 | 44,3 | 754,8 | 1557,2 |
| 30 | 13,8 | 79,5 | 110,6 | 1823,2 | 4054 |
| 50 | 25,1 | 176,5 | 154,5 | 4833,8 | 8971,8 |

Tablo 5.2: MOC-CDS algoritması benzetim sonuçları.

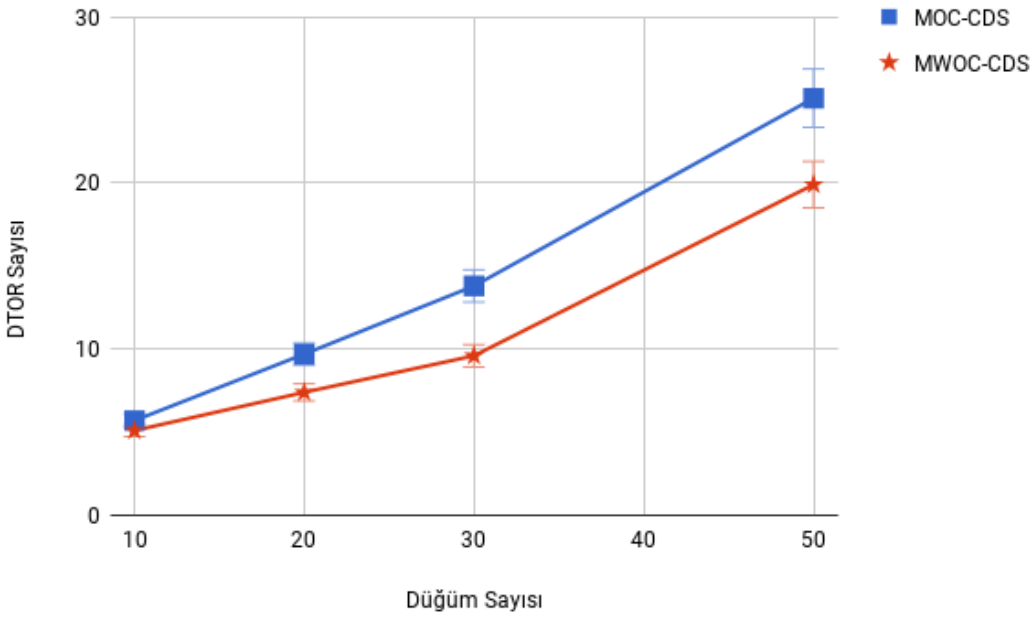
| Çizgedeki Düğüm Sayısı | Toplam DTOR Sayısı | Toplam CDS Ağırlığı | Toplam 2-kenar Sayısı | Toplam En Kısa Yol Kenar Sayısı | Toplam En Kısa Yol Ağırlığı |
|---|---|---|---|---|---|
| 10 | 5,1 | 22,4 | 13,4 | 208,6 | 516,6 |
| 20 | 7,4 | 29,9 | 35,4 | 846,2 | 1993,2 |
| 30 | 9,6 | 40,1 | 82,8 | 1938 | 3904,2 |
| 50 | 19,9 | 101,4 | 125,4 | 4866 | 7599 |

Şekil 5.8'de algoritmaların oluşturduğu CDS'lerin içerisindeki düğümlerin toplam ağırlıklarının çizgede bulunan düğüm sayısına göre grafiği verilmiştir. Sonuçlara göre MWOC-CDS algoritması, MOC-CDS algoritmasına göre daha düşük ağırlıklı kümeler oluşturmaktadır. Düğüm sayısı 10 olan çizgelerde %30, 20 olan çizgelerde %44, 30 olan çizgelerde %49 ve 50 olan çizgelerde ise %42 azalma gözlemlenmiştir. Küme ağırlıkları arasındaki azalma %68'lere kadar varabilmektedir.



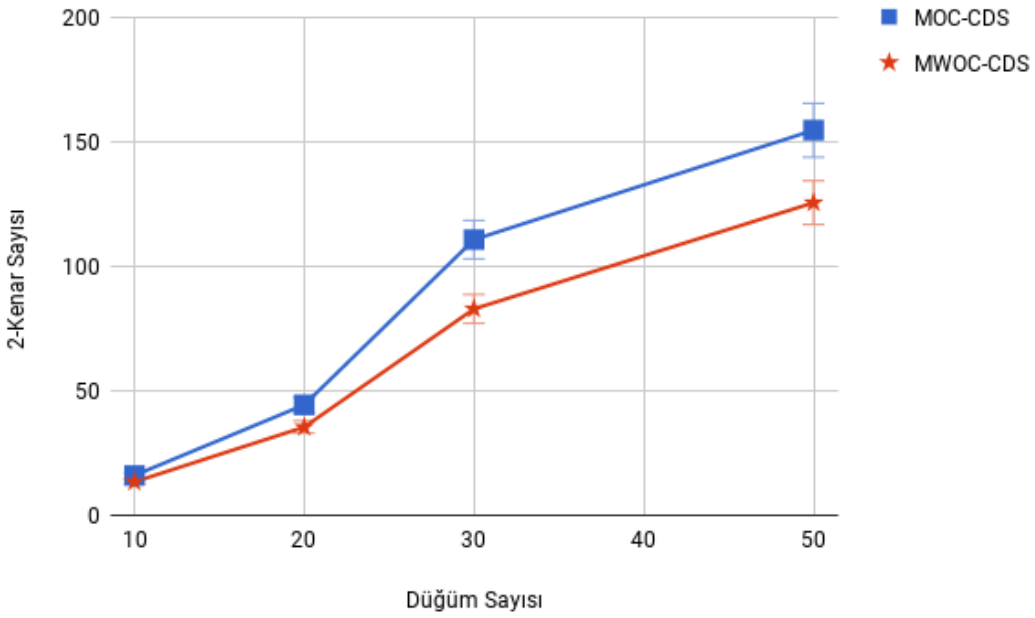

Şekil 5.8: MOC-CDS ve MWOC-CDS için toplam ağırlık – düğüm sayısı grafiği.

Şekil 5.9'da algoritmaların oluşturduğu CDS'lerin içerisinde bulunan düğüm sayılarının çizgede bulunan düğüm sayısına göre grafiği verilmiştir. Sonuçlara göre MWOC-CDS algoritması MOC-CDS algoritmasına göre daha düşük sayıda düğüm içeren CDS'ler oluşturmaktadır. Düğüm sayısı 10 olan çizgelerde %10, 20 olan çizgelerde %23, 30 olan çizgelerde %30 ve 50 olan çizgelerde ise %20 daha az düğüm seçilerek CDS oluşturulmuştur. Düğüm sayıları arasındaki fark %53'lere ulaşabilmektedir.



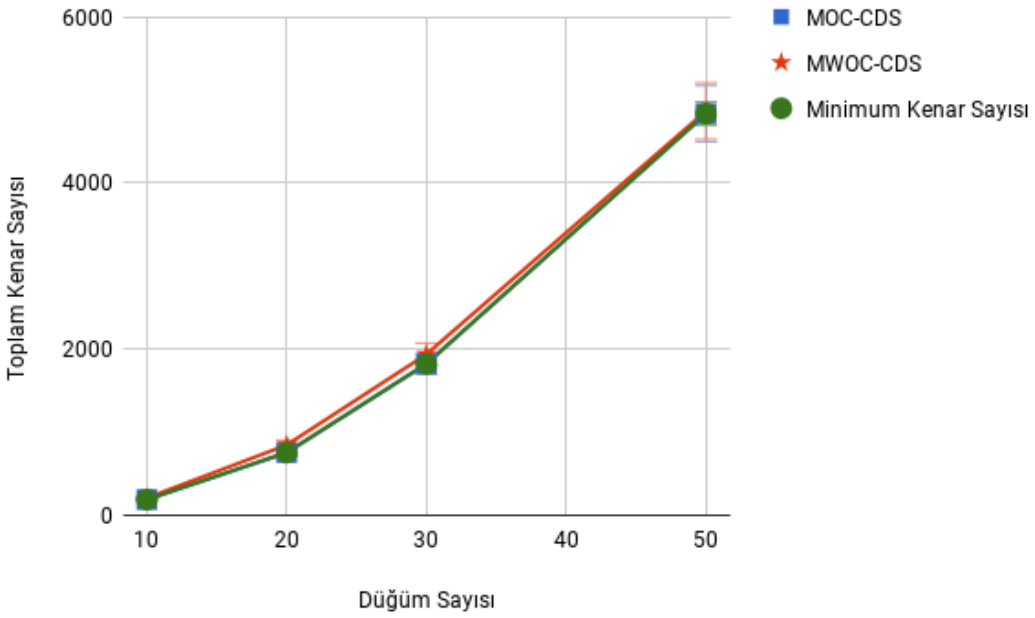

Şekil 5.9: MOC-CDS ve MWOC-CDS için DTOR sayısı – düğüm sayısı grafiği.

Şekil 5.10'da algoritmalar tarafından oluşturulan CDS'lerin içinde bulunan düğümlerin, aralarında köprü görevi gördüğü ikili düğüm sayılarının, çizgedeki düğüm sayısına göre değişimi verilmiştir. Bu noktada MOC-CDS algoritması daha iyi sonuç vermektedir. MWOC-CDS algoritması 10 düğümlü çizgelerde %16, 20 düğümlü çizgelerde %20, 30 düğümlü çizgelerde %25 ve 50 düğümlü çizgelerde ise %18 daha az düğüm ikilisi bağlamıştır. MOC-CDS ile MWOC-CDS algoritmalarının aralarındaki fark %34'lere varmaktadır.



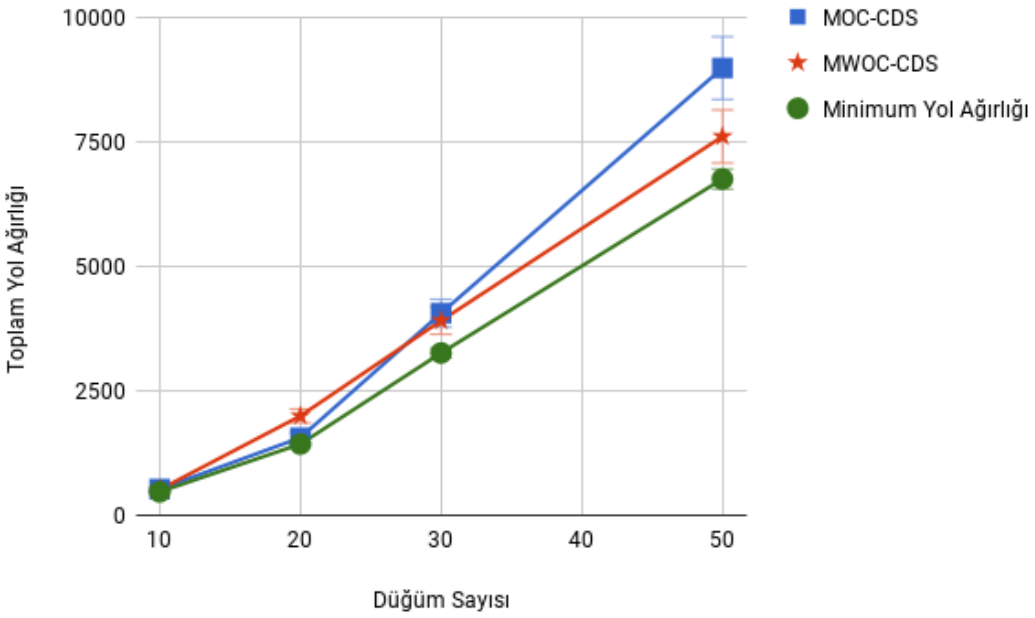

Şekil 5.10: MOC-CDS ve MWOC-CDS için 2-Kenar sayısı – düğüm sayısı grafiği.

Şekil 5.11'de çizge içerisindeki düğümlerin, algoritmalar tarafından oluşturulan CDS üzerinden, diğer düğümlere olan en kısa yolları üzerindeki kenar sayılarının toplamının ve optimum en kısa yollar üzerindeki kenar sayılarının toplamının, çizgedeki düğüm sayısına göre değişimi verilmiştir. Optimum en kısa yollar üzerindeki kenar sayıları, sadece CDS içerisindeki düğümler kullanılarak değil, çizgedeki tüm düğümler kullanılarak hesaplanmıştır. Sonuçlarda görüldüğü üzere MOC-CDS algoritması optimum sonuçları üretebilmektedir. MWOC-CDS algoritması ise 10 düğümlü çizgelerde %10, 20 düğümlü çizgelerde %12, 30 düğümlü çizgelerde %6 ve 50 düğümlü çizgelerde %3 daha fazla kenar ile düğümler arası bağlantı sağlamaktadır. MWOC-CDS algoritması bazı çizgeler üzerinde optimum sonuçlardan %1 daha fazla düğüm kullanarak düğümleri birbirlerine bağlayabilmektedir.




Şekil 5.11: MOC-CDS ve MWOC-CDS için toplam kenar sayısı – düğüm sayısı grafiği.

Şekil 5.12'de çizge içerisindeki düğümlerin, algoritmaların ürettiği CDS üzerinden, diğer düğümlere olan en küçük ağırlıklı yolların toplam ağırlıkları ve optimum en küçük ağırlıklı yolların toplam ağırlıkları verilmiştir. Sonuçlara bakıldığında MWOC-CDS algoritmasının optimum sonuçlara daha yakın olduğu gözlenmektedir. MWOC-CDS algoritması MOC-CDS algoritmasına göre 10, 20, 30 ve 50 düğümlü çizgelerde sırasıyla %1 daha az, %27 daha fazla , %3 daha az ve %15 daha az, optimum sonuçlara göre ise sırasıyla %9, %39, %19 ve %12 daha fazla toplam ağırlık ile düğümler arasında bağlantı kurabilmektedir. MWOC-CDS ile MOC-CDS algoritmaları arasındaki fark %42'lere kadar çıkabilmektedir. MWOC-CDS ile optimum sonuçlar arasındaki fark ise %1'lere kadar inebilmektedir.




Şekil 5.12: MOC-CDS ve MWOC-CDS için toplam yol ağırlığı – düğüm sayısı grafiği.

Sonuç olarak yeni bir minimal ağırlıklı ve yönlendirme maliyetli bağlı baskın küme (MWOC-CDS) algoritması geliştirilmiştir. Geliştirilen algoritma MOC-CDS algoritması ile karşılaştırılmıştır. Benzetimlerde oluşan kümelerin, toplam ağırlıkları, düğüm sayıları ve küme içerisinden geçen en kısa ağırlıklı yolların toplam ağırlıkları önemli ölçüde düşerken, küme içerisinden geçen en kısa yolların azaldığı gözlemlenmiştir.

# 6. WANG, ASYNSET-ASYNTREE ve CN-MWCDS

Tezin bu bölümünde çift fazlı minimum ağırlıklı bağlı baskın küme (MWCDS) algoritmaları olan Wang ve diğ. (2006)'nin geliştirdiği MWCDS algoritması, ASYNSET-ASYNTREE (Dagdeviren ve diğ. 2015) algoritması ve tez kapsamında geliştirilen CN-MWCDS algoritması incelenmiştir.

## 6.1 WANG Algoritması

Wang ve diğ. (2006)'nin önerdikleri algoritma çift fazlı minimum ağırlıklı bağlı baskın küme oluşturma algoritmasıdır. Algoritmanın ilk fazında minimum bağlı baskın küme oluşturulması hedeflenmiştir. İkinci fazında ise ilk fazda seçilen düğümler arasında sanal bir ağ oluşturularak düğümler birbirlerine bağlanmaktadır.

Algoritmanın ilk fazında düğümlerin ağırlıkları IDLE koşularının ağırlıkları içerisinde en küçük değere sahip ise, PDTOR olarak seçilmektedir. PDTOR olarak seçilen düğümler iki-kenar uzaklıktaki komşu bilgilerini toplamaktadır. PDTOR düğümler topladıkları bu 2-kenar uzaklıktaki çizge üzerinde yerel açgözlü bir küme kapsama algoritması çalıştırmaktadır. Çalıştırılan algoritmada kapsanacak küme PDTOR olan düğüm ve PDTOR olan düğümün IDLE komşuları olmaktadır. Algoritmanın amacı ise bu kümeyi kapsayan minimum ağırlıklı bir küme bulmaktır. Bulunan bu kümeye GRDY kümesi denmektedir. GRDY kümesi bulunurken düğümlerin ağırlıklarının kapsanacak kümedeki komşu sayılarına oranı ağırlık oranı olarak kullanılmaktadır. Her turda ağırlık oranı en düşük olan düğüm seçilmektedir. Kapsanacak kümedeki tüm düğümler kapsandığında yerel algoritma sona ermektedir ve GRDY kümesi oluşmaktadır. Ardından GRDY kümesindeki düğümlerin toplam ağırlığı, PDTOR düğümün ağırlığından küçük ise GRDY kümesindeki düğümler DTOR olarak seçilmektedir. Aksi takdirde PDTOR olarak seçilen düğüm DTOR olarak seçilmektedir.

Algoritmanın ikinci fazında DTOR olarak seçilen düğümleri birbirlerine bağlamak için sanal bir ağ oluşturulmaktadır. DTOR düğümler yeni oluşturulan ağda

köşe görevi görmektedir. DTEE düğümler ise ağırlıklı kenar görevi görmektedirler. Sanal ağ oluşturulduktan sonra DTOR düğümler arasında MST algoritması çalıştırılmaktadır ve seçilen sanal kenarlar içerisindeki düğümler DTOR olarak seçilerek bağlı baskın küme oluşturulmaktadır.

### 6.1.1 WANG Algoritmasının Birinci Fazı

---

**Algoritma 3:** *m* düğümü için WANG algoritmasının birinci fazı

---

**Tur başladığında** IDLE komşuları içerisinde en düşük ağırlığa sahip ve IDLE durumunda ise durumunu PDTOR yap. $\text{ItryDominator}_m$ mesajını tüm komşularına gönder. 2-kenar uzaklıktaki düğüm bilgilerini topla.

**$\text{ItryDominator}_u$ mesajını aldığında durum IDLE ise** durumunu DTEE yap, $\text{IamDominatee}_m$ mesajını tüm komşularına gönder ve komşu bilgilerini ağırlıkları ile beraber *u* düğümüne gönder.

**$\text{IamDominatee}_u$ mesajını aldığında** *u* düğümünün durumunu DTEE olarak değiştir.

**Tüm IDLE komşularından komşu bilgilerini topladığında** 2-kenar uzaklıktaki düğümler kendisi ve komşuları içerisinde, kendisini ve tüm IDLE komşularını kapsayacak şekilde yerel açgözlü küme kapsama algoritmasını çalıştır ve $\text{GRDY}_m$ kümesini bul. $\text{GRDY}_m$ kümesinin toplam ağırlığı kendi ağırlığından küçük ise kendi durumunu DTEE yap ve $\text{GRDY}_m$ kümesi içerisindeki her *w* düğümüne YouAreDominator(*w*) mesajını gönder. Aksi takdirde kendi durumunu DTOR yap ve $\text{IamDominator}_m$ mesajını tüm komşularına gönder.

**YouAreDominator(*m*) mesajını aldığında** kendi durumunu DTOR yap ve $\text{IamDominator}_m$ mesajını tüm komşularına gönder.

**$\text{IamDominator}_u$ mesajını aldığında** *u* düğümünün durumunu DTOR yap. Kendi durumu IDLE ise durumunu DTEE yap ve $\text{IamDominatee}_m$ mesajını tüm komşularına gönder.

---

Algoritmanın başlangıcında tüm düğümler kendi ağırlıklarını, komşularını ve komşularının ağırlıklarını bilmektedirler. Algoritma 3'te ***m*** düğümü için WANG algoritmasının birinci fazının işleyişi verilmiştir.

Algoritma 3'te verilen işleyişe göre algoritma başladığında IDLE ***m*** düğümü kendi ağırlığı ile komşularının ağırlıklarını kıyaslamaktadır. Kendi ağırlığı komşularının ağırlıkları içerisinde en küçük ağırlık ise ***m*** düğümü kendi durumunu PDTOR yapmaktadır ve komşularını haberdar etmek için $\boldsymbol{ItryDominator_m}$ mesajını göndermektedir. Ağırlıklarda eşitlik yaşanması durumunda ID'si küçük olan düğüm PDTOR olarak seçilmektedir.

IDLE bir ***m*** düğümü bir ***u*** komşusundan $\boldsymbol{ItryDominator_u}$ mesajı aldığında kendi durumunu DTEE olarak değiştirmektedir ve komşularına DTEE olduğu bilgisini vermek için $\boldsymbol{IamDominatee_m}$ mesajını göndermektedir. Daha sonra ***u*** düğümünün yerel kapsama algoritmasını çalıştırabilmesi için komşu bilgilerini ağırlıkları ile birlikte ***u*** düğümüne göndermektedir.

PDTOR bir ***m*** düğümü tüm IDLE komşularından komşuluk bilgilerini topladıktan sonra yerel açgözlü bir kapsama algoritması çalıştırmaktadır. Bu algoritmanın amacı ***m*** düğümü ve IDLE komşularını kapsayan minimum ağırlıklı bir küme bulmaktır. Kapsayan kümeyi bulurken seçilen düğümler, düğümlerin ağırlıklarının ***m*** ve ***m***'nin IDLE komşularının kaç tanesine komşu olduklarına oranı kullanılarak bulunur. Her turda en küçük orana sahip olan düğüm seçilmektedir, en küçük orana sahip birden fazla düğüm var ise ID'si küçük olan düğüm seçilmektedir. ***m*** ve ***m***'nin tüm IDLE komşuları kapsandığında yerel kapsama algoritması sonlanmaktadır. Algoritmanın sonucunda oluşan kapsayan küme $\boldsymbol{GRDY_m}$ şeklinde gösterilmektedir. ***m*** düğümü ile $\boldsymbol{GRDY_m}$ kümesinin ortak yanı, ikisinin de ***m*** ve ***m***'nin IDLE komşularını kapsamasıdır. Bu durumda DTOR olarak ya ***m*** ya da $\boldsymbol{GRDY_m}$ seçilmektedir. Hangisinin DTOR olarak seçileceğine karar vermek için ***m*** düğümünün ağırlığı ile $\boldsymbol{GDRY_m}$ kümesindeki düğümlerin toplam ağırlığı kıyaslanmaktadır. ***m*** düğümünün ağırlığı daha küçük ise ***m*** düğümü DTOR olarak seçilmektedir ve $\boldsymbol{IamDominator_m}$ mesajını tüm komşularına bilgi vermek için göndermektedir. $\boldsymbol{GRDY_m}$ kümesinin toplam ağırlığı daha düşük ise ***m*** düğümü kendi durumunu DTEE yapmaktadır ve $\boldsymbol{GRDY_m}$ kümesinin içerisindeki her ***w*** düğümü için ***YouAreDominator(w) w*** düğümüne iletilmek üzere göndermektedir.

***m*** düğümü ***YouAreDominator(m)*** mesajını aldığında kendi durumunu DTOR olarak değiştirmektedir ve $\boldsymbol{IamDominator_m}$ mesajını tüm komşularına bilgi vermek için göndermektedir.

$\boldsymbol{IamDominator_u}$ mesajı alan ***m*** düğümü ***u*** düğümünün durumunu DTOR olarak değiştirmektedir. ***m*** düğümünün durumu IDLE ise durumunu DTEE olarak değiştirmekte ve komşularına bilgi vermek için $\boldsymbol{IamDominatee_m}$ mesajını göndermektedir.

$\boldsymbol{IamDominatee_u}$ mesajını alan bir ***m*** düğümü ***u*** düğümünün durumunu DTEE olarak değiştirmektedir.

Çizge üzerinde IDLE düğüm kalmayıncaya kadar turlar tekrar çalışmaktadır. Çizge üzerinde IDLE düğüm kalmadığında algoritmanın birinci fazı sonlanmaktadır ve ikinci fazı başlatılmaktadır.

### 6.1.2 WANG Algoritmasının İkinci Fazı

Algoritmanın birinci fazı sonlandığında çizgede DTOR ve DTEE düğümler bulunmaktadır ve bu DTOR düğümler baskın bir küme oluşturmaktadır. Bağlı olmayan baskın kümeyi bağlı hale getirmek için DTOR düğümler arasında sanal bir ağ oluşturulmaktadır. Elde edilen sanal ağda DTOR düğümler köşe, DTEE düğümler ise kenarları oluşturmaktadır. Bir kenarın ağırlığı o kenarı oluşturan DTEE düğümlerin toplam ağırlığıdır. Sonuç olarak kenar ağırlıklı sanal bir ağ oluşturulmaktadır ve ardından oluşturulan sanal ağ üzerinde dağıtık bir MST algoritması çalıştırılarak baskın küme bağlı hale getirilmektedir. Algoritma 4'te WANG algoritmasının ikinci fazının ***m*** düğümü için işleyişi verilmiştir.

**Algoritma 4:** *m* düğümü için WANG algoritmasının ikinci fazı

**Algoritma başladığında durum DTEE ise** $\text{OneHopDominatorList}_m$ mesajını tüm komşularına gönder.

**$\text{OneHopDominatorList}_u$ mesajını aldığında** alınan mesaj içerisindeki her DTOR düğüm için $D^2_m$ kümesini güncelle.

**Tüm OneHopDominatorList mesajlarını aldığında durum DTEE ise** $\text{TwoHopDominatorList}_m$ mesajını DTOR komşularına gönder.

**$\text{TwoHopDominatorList}_u$ mesajını aldığında** alınan mesaj içerisindeki her DTOR düğüm için $D^3_m$ kümesini güncelle.

**Tüm TwoHopDominatorList mesajlarını aldığında** oluşan sanal ağ üzerinde dağıtık bir MST algoritması çalıştır.

Algoritma 4'e göre algoritmanın başlangıcında tüm DTEE düğümler DTOR komşu düğümlerini ***OneHopDominatorList*** mesajı olarak tüm komşularına göndermektedir.

Bir ***m*** düğümü $\boldsymbol{OneHopDominatorList_u}$ mesajı aldığında, 2-kenar uzaklıktaki DTOR düğümlerin ve yollarının tutulduğu $\boldsymbol{D^2_m}$ kümesini güncellemektedir. Güncelleme işlemi mesajın içerisindeki DTOR düğümlerin m düğümü ile arasındaki yolun ağırlığına bakılarak yapılmaktadır. Mesaj içerisinde bulunan ve ***m*** düğümünün komşuları içerisinde bulunmayan her DTOR ***w*** düğümü için $\boldsymbol{D^2_m}$ kümesine bakılmaktadır. $\boldsymbol{D^2_m}$ kümesinin içerisinde ***w*** düğümü yok ise ***w*** düğümü $\boldsymbol{D^2_m}$ kümesine eklenmektedir ve yolu ***u*** düğümü olarak kayıt edilmektedir. ***w*** düğümü $\boldsymbol{D^2_m}$ kümesinin içerisinde var ise, ***w*** düğümünün yol ağırlığı ile ***u*** düğümünün ağırlığı kıyaslanmaktadır. ***u*** düğümünün ağırlığı daha düşük ise ***w*** düğümünün yolu ***u*** düğümü olarak değiştirilmektedir.

DTEE bir ***m*** düğümü tüm DTEE komşularından ***OneHopDominatorList*** mesajlarını aldığında, 2-kenar uzaklıktaki tüm DTOR düğümlerini ve onlara olan en kısa uzaklıklarını $\boldsymbol{D^2_m}$ kümesinde toplamış olmaktadır. Toplanan bu $\boldsymbol{D^2_m}$ kümesini

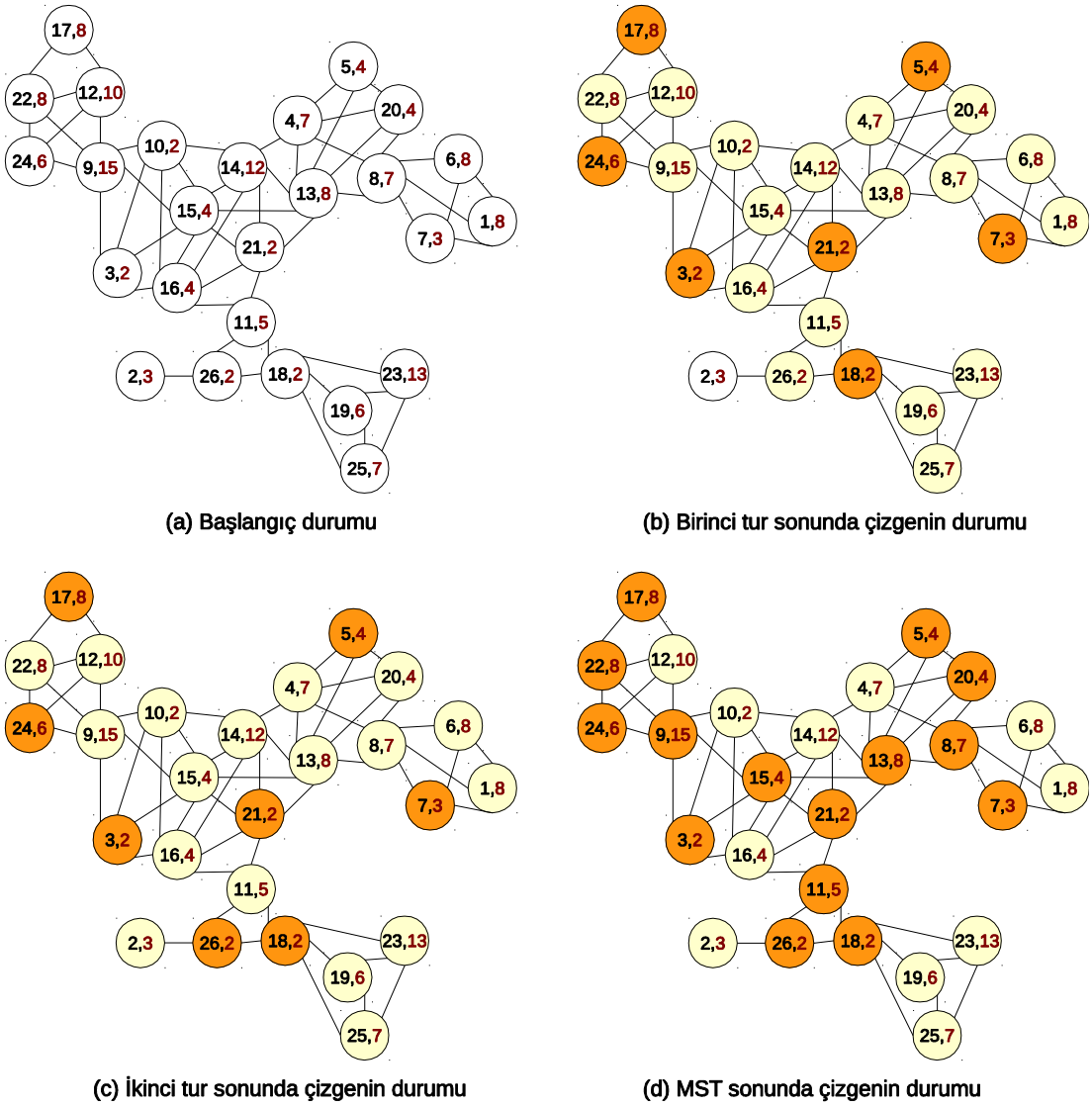
tüm DTOR komşularını bilgilendirmek için $TwoHopDominatorList_m$ mesajı ile göndermektedir.

DTOR bir $m$ düğümü $TwoHopDominatorList_u$ mesajı aldığında aldığı $D^2_u$ kümesi ile $D^3_m$ kümesini güncellemektedir. Güncelleme yaparken $D^2_u$ kümesinin içerisinde bulunan ve $m$ kümesinin komşuları içerisinde bulunmayan her $w$ düğümü için $D^2_m$ ve 3-kenar uzaklıktaki DTOR düğümlerin ve yol bilgilerinin tutulduğu $D^3_m$ kümelerine bakılmaktadır. $w$ düğümü $D^2_m$ içerisinde varsa ve $D^2_m$ içerisindeki yol ağırlığı, $u$ düğümünün ağırlığın ile $D^2_u$ içerisindeki $w$ düğümünün yol ağırlığının toplamından daha büyük ise $w$ düğümü $D^2_m$ kümesinden çıkartılmaktadır ve $D^3_m$ kümesine eklenerek yol bilgisi $u$ düğümü ile $D^2_u$ içerisindeki $w$ düğümünün yolunun toplamı olarak değiştirilmektedir. $w$ düğümü $D^2_m$ kümesi içerisinde yoksa ve $D^3_m$ kümesinin içerisinde var ise yol ağırlıklarına bakılır. $u$ düğümünün ağırlığı ile $D^2_u$ kümesindeki $w$ düğümünün yol ağırlığının toplamı, $D^3_m$ kümesindeki $w$ düğümünün yol ağırlığından küçük ise $D^3_m$ kümesindeki $w$ düğümünün yol bilgisi $u$ düğümü ile $D^2_u$ kümesindeki $w$ düğümünün yolunun toplamı olarak değiştirilmektedir. $w$ düğümü hem $D^2_m$ hem de $D^3_m$ kümeleri içerisinde yoksa $w$ düğümü $D^3_m$ kümesine eklenerek yol bilgisi $u$ düğümü ile $D^2_u$ içerisindeki $w$ düğümünün yolunun toplamı olarak kayıt edilmektedir.

Tüm DTEE komşularından $TwoHopDominatorList$ mesajlarını alan DTOR bir $m$ düğümü sanal ağ üzerindeki komşularını ve onlar ile arasındaki en kısa yolları bilmektedir. Komşusu olan DTOR düğümler $D^1_m$ kümesinde, aralarında bir DTEE düğüm bulunan DTOR düğümler $D^2_m$ kümesinde ve aralarında iki DTEE düğüm bulunan DTOR düğümler $D^3_m$ kümesinde bulunmaktadır. $D^1_m$ kümesi içerisindeki DTOR düğümler ile $m$ düğümü arasındaki yolların ağırlıkları 0 olmaktadır. Bu üç küme içerisindeki DTOR düğümler $m$ düğümünün sanal komşuları olmaktadır ve yol ağırlıkları kümeler içerisinde tutulmaktadır. Bu bilgiler kullanılarak sanal ağ üzerinde dağıtık bir MST algoritması çalıştırılmaktadır. MST algoritmasında seçilen yolları oluşturan DTEE düğümler durumlarını DTOR olarak değiştirmektedir ve sonuç olarak bağlı baskın küme oluşturulmaktadır.

### 6.1.3 WANG Algoritmasının Örnek Üzerinde Gösterimi

Şekil 6.1'de WANG algoritmasının düğüm-ağırlıklı bir çizge üzerindeki adımları verilmiştir. Şekilde beyaz ile gösterilen düğümler IDLE, turuncu ile gösterilenler DTOR ve sarı ile gösterilenler DTEE düğümlerdir.




Şekil 6.1: WANG algoritmasının örnek işleyişi.

Algoritmanın başlangıç durumu Şekil 6.1 (a)'da gösterilmiştir. Algoritmanın birinci turunda tüm beyaz komşularından daha düşük ağırlığa sahip olan beyaz düğümler kendi durumlarını PDTOR olarak değiştirerek ***ItryDominator*** mesajlarını tüm komşularına göndermektedir. Bu düğümler sırasıyla ***3, 5, 7, 17, 18, 21*** ve ***24*** numaralı düğümlerdir. ***ItryDominator*** mesajını alan ***1, 4, 6, 8, 9, 10, 11, 12, 13, 14,***

***15, 16, 19, 20, 22, 23, 25*** ve ***26*** numaralı düğümler IDLE durumunda oldukları için kendi durumlarını DTEE olarak değiştirmektedir, komşuluk bilgilerini ağırlıkları ile birlikte ***ItryDominator*** mesajını aldıkları düğüme göndermektedir ve ***IamDominatee*** mesajlarını tüm komşularına göndermektedir. PDTOR olarak seçilen düğümler 2-kenar uzaklıktaki düğüm bilgilerini topladıktan sonra yerel kapsama algoritmasını çalıştırmaktadır. Çalıştırılan algoritmalarının sonucunda hiçbir düğümün ***GRDY*** kümesinin toplam ağırlığı kendi ağırlığından küçük çıkmadığı için tüm PDTOR düğümler kendi durumlarını DTOR olarak değiştirmektedir ve ***IamDominator*** mesajlarını tüm komşularına göndermektedir. Örneğin ***3*** numaralı düğüm komşuları olan ***9, 10, 15*** ve ***16*** numaralı düğümler arasında minimum ağırlığa sahip olduğu için PDTOR olarak seçilmiş ve 2-kenar uzaklıktaki düğüm bilgilerini toplamıştır. Bu düğümler ***11, 12, 13, 14, 21, 22*** ve ***24*** numaralı düğümlerdir. ***3*** numaralı düğüm bu bilgileri topladıktan sonra kendisi komşuları ve 2-kenar uzaklıktaki düğüm bilgilerini kullanarak tur başlangıcında beyaz olan ***3, 11, 12, 13, 14, 21, 24*** numaralı düğümleri kapsayan minimum ağırlıktaki kümeyi bulmak için yerel küme kapsama algoritmasını çalıştırmaktadır. Algoritmanın sonucunda $\boldsymbol{GRDY_3}$ kümesi ***{3}*** şeklinde olmaktadır. ***3*** numaralı düğümün ağırlığı $\boldsymbol{GRDY_3}$ kümesinin toplam ağırlığından küçük veya eşit olduğundan ***3*** numaralı düğüm durumunu DTOR olarak değiştirmektedir. Birinci turun sonunda çizge Şekil 6.1 (b)'deki halini almaktadır.

İkinci turda beyaz komşuları içerisinde minimum ağırlığa sahip olan tek beyaz düğüm ***2*** numaralı düğümdür. Dolayısıyla sadece ***2*** numaralı düğüm kendisinin durumunu PDTOR olarak işaretlemekte ve $\boldsymbol{ItryDominator_2}$ mesajını göndermektedir. ***2*** numaralı düğümden $\boldsymbol{ItryDominator_2}$ mesajını alan ***26*** numaralı düğüm IDLE olmadığı için herhangi bir şey yapmamaktadır. ***2*** numaralı düğümün IDLE komşusu olmadığı için 2-kenar uzaklıktaki düğüm bilgilerini toplamasına gerek yoktur ve sadece kendisi ve komşuları içerisinden tur başlangıcında beyaz olan ***{2}*** kümesini kapsayan minimum ağırlıklı kümeyi bulmak için yerel kapsama algoritmasını çalıştırmaktadır. Algoritma başladığında ***2*** numaralı düğümün ağırlık oranı, ***2*** numaralı düğümün ağırlığı olan ***3***'ün kapsanacak kümedeki kapsadığı elman sayısı olan 1'e oranıdır. Dolayısıyla ***2*** numaralı düğümün ağırlık oranı ***3/1=3***'tür. Aynı ağırlık oranı hesaplaması ***26*** numaralı düğüm için de yapılır ve ***2/1=2*** olarak bulunur. En küçük ağırlık oranına sahip olan ***26*** numaralı düğüm $\boldsymbol{GRDY_2}$ kümesine dahil edilir. Kapsanması gereken tüm beyaz düğümler kapsandığı için yerel

algoritma sonlandırılmaktadır. $GRDY_2$ kümesinin toplam ağırlığı ***2*** numaralı düğümün ağırlığından küçük olduğu için ***2*** numaralı düğüm kendi durumunu DTEE olarak değiştirmektedir ve $GRDY_2$ içerisinde bulunan ***26*** numaralı düğüme ***YouAreDominator(26)*** mesajını göndermektedir. ***YouAreDominator(26)*** mesajını alan ***26*** numaralı düğüm kendi durumunu DTOR olarak değiştirmektedir ve $IamDominator_{26}$ mesajını tüm komşularına göndermektedir.

İkinci turun sonunda çizgede IDLE düğüm kalmadığı için algoritmanın birinci fazı sonlanmaktadır ve ikinci fazı başlamaktadır. Çizgenin birinci faz sonrasındaki durumu Şekil 6.1(c)'de gösterilmektedir.

İkinci faz başladığında DTEE olan ***1, 2, 4, 6, 8, 9, 10, 11, 12, 13, 14, 15, 16, 19, 20, 22, 23*** ve ***25*** numaralı düğümler komşusu olan DTOR düğümleri tüm komşularına ***OneHopDominatorList*** mesajı ile göndermektedir. Örneğin ***12*** numaralı düğüm ***{17, 24}*** kümesini ***OneHopDominatorList*** mesajı ile tüm komşularına göndermektedir.

***OneHopDominatorList*** mesajını alan düğümler kendi $D^2$ kümelerini güncellemektedir. Örneğin ***12*** numaralı düğümden $OneHopDominatorList_{12}$ mesajını alan ***9*** numaralı düğüm, aldığı ***{17, 24}*** kümesindeki her düğüm için $D^2_9$ kümesini güncellemektedir. ***17*** numaralı düğümü, kendi komşuları içerisinde bulunmadığı için $D^2_9$ kümesine yol bilgisi ***12*** numaralı düğüm olacak şekilde eklemektedir ve $D^2_9 = \{(17,12)\}$ olmaktadır. ***24*** numaralı düğüm ***9*** numaralı düğümün zaten komşusu olduğu için $D^2_9$ kümesine eklenmemektedir. Ardından ***15*** numaralı düğümden ***{3, 21}*** mesajını aldığında, ***3*** numaralı düğüm zaten komşusu olduğu için eklenmemekte ve ***21*** numaralı düğüm komşusu olmadığı ve $D^2_9$ kümesinde bulunmadığı için yol bilgisi ***15*** olacak şekilde eklenmektedir. ***15*** numaralı düğümden alınan mesaj sonucunda $D^2_9 = \{(17, 12), (21, 15)\}$ olmaktadır. Daha sonra ***9*** numaralı düğüm ***22*** numaralı düğümden ***{17, 24}*** kümesini aldığında ***24*** numaralı düğüm ***9*** numaralı düğümün komşusu olduğu için $D^2_9$ kümesine eklememektedir. ***17*** numaralı düğüm ise $D^2_9$ kümesinde zaten olduğu için eklenmemektedir. Ancak $D^2_9$ kümesinde ***17*** numaralı düğümün yolu ***12*** numaralı düğümdür ve ağırlığı 10'dur. ***22*** numaralı düğümün ağırlığı 8 10'dan daha küçük olduğundan $D^2_9$ kümesindeki ***17*** numaralı düğümün yol bilgisi ***22*** olacak şekilde değiştirilmektedir. Sonuç olarak $D^2_9=\{(17,22),(21,15)\}$ olmaktadır.

Tüm düğümler tüm DTEE komşularından ***OneHopDominatorList*** mesajlarını aldıktan sonra bazı $D^2$ kümeleri aşağıdaki gibi olmaktadır.

$D^2_{11} = \{(3, 16)\}$ , $D^2_{16} = \{(18, 11), (26, 11)\}$

$D^2_{26} = \{(21, 11)\}$ , $D^2_{24} = \{(17, 22), (3, 9)\}$

Tüm DTEE komşularından ***OneHopDominatorList*** mesajlarını alan DTEE düğümler güncel $D^2$ kümelerini ***TwoHopDominatorList*** mesajı ile DTOR komşularına göndermektedir.

***TwoHopDominatorList*** mesajını alan DTOR düğümler kendi $D^3$ kümelerini güncellemektedir. Örneğin $TwoHopDominatorList_9$ mesajı ile $D^2_9=\{(17,22),(21,15)\}$ kümesini alan ***24*** numaralı düğüm $D^2_9$ kümesindeki her düğüm için $D^3_{24}$ kümesini güncellemektedir. ***17*** numaralı düğüm $D^2_{24}$ kümesinde bulunduğu için yol ağırlıkları kıyaslanmaktadır. ***17*** numaralı düğümün $D^2_{24}$ kümesindeki yolu ***22*** numaralı düğüm olduğundan yol ağırlığı ***22*** numaralı düğümün ağırlığına yani 8'e eşittir. ***9*** numaralı düğüm ile ***17*** numaralı düğümün $D^2_9$ kümesindeki yolu olan ***22*** numaralı düğümlerin toplam ağırlığı 15+8=23'e eşittir. $D^2_{24}$ kümesindeki yol ağırlığı daha küçük olduğu için ***17*** numaralı düğüm $D^3_{24}$ kümesine eklenmemektedir. ***21*** numaralı düğüm ise hem ***24*** numaralı düğümün komşusu olmadığından hem de $D^2_{24}$ kümesinde bulunmadığından $D^3_{24}$ kümesine yol bilgisi ***9*** ve ***15*** olacak şekilde eklenmektedir. Sonuç olarak $D^3_{24} = \{(21,9\text{-}15)\}$ olmaktadır. Tüm DTOR düğümler tüm DTEE komşularından ***TwoHopDominatorList*** mesajlarını aldıktan sonra DTOR düğümlerin $D^1$, $D^2$ ve $D^3$ kümeleri Tablo 6.1'de verilmiştir.

Tablo 6.1: Şekil 6.1 (c)'de verilen DTOR düğümlerin $D^1$, $D^2$ ve $D^3$ kümeleri.

| Düğüm ID'si | $D^1$ kümesi | $D^2$ Kümesi | $D^3$ Kümesi |
|---|---|---|---|
| 3 | - | (21,15), (24,9) | (17,9-22), (26,16-11), (18,16-11) |
| 5 | - | (21,13) | (7,20-8) |
| 7 | - | - | (5,8-20), (21,8-13) |
| 17 | - | (24,22) | (3,12-9) |
| 18 | 26 | (21,11) | (3,11-16) |
| 21 | - | (3,15), (18,11), (16,11), (5,13) | (7,13-8), (24,15-9) |
| 24 | - | (3,9), (17,22) | (21,9-15) |
| 26 | 18 | (21,11) | (3,11-16) |

Tüm DTOR düğümler $D^1$, $D^2$ ve $D^3$ kümelerini oluşturduktan sonra sanal ağ oluşmuş olmaktadır ve bu sanal ağ üzerinde dağıtık bir MST algoritması çalıştırılmaktadır. MST algoritması çalıştırıldıktan sonra algoritma tarafından seçilen sanal kenarlar üzerindeki tüm DTEE düğümler DTOR olarak seçilmiş bulunmaktadır. Bu düğümler sırasıyla ***8, 9, 11, 13, 15, 20*** ve ***22*** numaralı düğümlerdir. Sonuç olarak Şekil 6.1 (d)'de gösterilen bağlı baskın küme oluşmaktadır.

## 6.2 ASYNSET ve ASYNTREE Algoritmaları

Dagdeviren ve diğ. (2015)'nin önerdiği ASYNSET ve ASYNTREE algoritmaları bir minimum ağırlıklı bağlı baskın küme (MWCDS) oluşturma algoritmasıdır. Çift fazlı bir algoritmadır. Diğer çift fazlı algoritmalarda olduğu gibi ilk faz olan ASYNSET algoritmasında baskın küme oluşturulmakta ve ikinci faz olan ASYNTREE algoritmasında ise Steiner Ağacı problemine bir yaklaşım algoritması kullanılarak baskın küme bağlı baskın küme haline getirilmektedir.

ASYNSET ve ASYNTREE algoritmalarında turları yönetmek ve senkronizasyonu sağlamak için β-syncronizer kullanılmaktadır. β-syncronizer bir ağaç üzerinde çalışmaktadır. Yani β-syncronizer kullanılabilmesi için önceden oluşturulmuş bir ağaca ihtiyaç duyulmaktadır. Çizge içerisindeki tüm düğümler bu ağacın içerisinde olmalı ve ebeveyn çocuk bilgilerini bilmelidir. Kök düğüm turu başlatmak istediğinde çocuklarına tur başladı mesajını göndermektedir. Bu mesajı alan her çocuk kendi çocuklarına mesajı iletmektedir. Bu sayede tur tüm düğümlerde başlamış olmaktadır. Turun bitmesi için ise çocuklar OK mesajlarını ebeveynlerine göndermektedir. Tüm çocuklarından OK mesajını alan ebeveyn düğüm OK mesajını ebeveynine göndererek kök düğüme kadar iletilmektedir. Kök düğüm tüm çocuklarından OK mesajı aldığında turun bittiğini anlamaktadır ve yeni turu başlatabilmektedir.

Algoritmalarda düğümlerin ağırlıkları düğümlerin enerji seviyeleri ile ters orantılıdır. ASYNSET algoritması, düğümlerin ağırlıkları ve komşularının ağırlıklarını kullanarak hesapladığı ağırlık oranına göre DTOR düğümleri seçen minimum ağırlıklı baskın küme algoritmasıdır. $m$ düğümünün ağırlık oranı $wr_m$ ile

gösterilmektedir ve $wr_m = w_m / \sum w_u \mid u \in \Gamma_m \wedge state_u = IDLE$ şeklinde hesaplanmaktadır. $w_m$, $m$ düğümünün ağırlığını, $\Gamma_m$, $m$ düğümünün komşularını $state_u$ ise $u$ düğümünün durumunu göstermektedir. Sonuç olarak $wr_m$ ağırlık oranı $m$ düğümünün ağırlığının ($w_m$) $m$ düğümünün IDLE komşularının ağırlıklarının toplamına ($\sum w_u \mid u \in \Gamma_m \wedge state_u = IDLE$) oranıdır. Her turda ağırlık oranı 2-kenar uzaklıktaki komşularının ağırlık oranından daha küçük olan düğüm DTOR olarak seçilmektedir ve komşuları DTEE durumuna geçmektedir. Çizgede hiç IDLE düğüm kalmayıncaya kadar turlar devam etmektedir.

ASYNTREE algoritması bir Steiner Ağacı problemine bir yaklaşım algoritmasıdır. Algoritmada her turda bir düğüm seçilmektedir ve seçme işlemine kök düğüm karar vermektedir. Kök düğüm ve diğer düğümler seçme işlemini gerçekleştirirken yine ağırlık oranlarını kullanmaktadır. Her DTEE düğüm kendi ağırlık oranını hesaplayabilmektedir. DTEE bir $m$ düğümünün birden fazla komşu ağacı var ise $wr_m = w_m / |\Gamma^{tree}_m|$ şeklinde hesaplamaktadır. $\Gamma^{tree}_m$ komşu ağaçlardan oluşan bir kümedir. $m$ düğümünün sadece bir komşu ağacı var ise $m$ düğümü bir DTEE komşusuyla birleşerek ağırlık oranı hesaplamaktadır. Ağırlık oranı en küçük olacak şekilde bir DTEE komşusu seçmektedir. $Wr_m = (w_m + w_u) / |\Gamma^{tree}_m \, U \, \Gamma^{tree}_u|$ şeklinde hesaplanmaktadır. Her turda β ağacının kök düğümü en küçük ağırlık oranına sahip olan DTEE düğümünü DTOR olarak seçmektedir ve DTOR olan düğüm kendisini kök düğüm yaparak komşu ağaçlarını birleştirmektedir. Çizgede tek bir ağaç kalana kadar turlar devam etmektedir. Çizgede bir ağaç kaldığında algoritma sonlanmaktadır ve bağlı baskın küme oluşturulmaktadır.

### 6.2.1 ASYNSET Algoritması

Algoritmanın başlangıcında her düğüm komşularını ($\Gamma$), enerji seviyelerini ($e$), β-ağacındaki çocuklarını ($\Phi$) ve ebeveynini ($parent_s$) bilmektedir. Özet olarak her düğüm başlangıçta ağırlık oranlarını hesaplamak için komşularına enerji seviyelerini ***MY_ENERGY(e)*** mesajı ile göndermektedir. Komşularının enerjisini öğrenen düğümler ağırlık oranlarını ($wr$) hesaplamaktadır ve ***MY_WRATIO(wr)*** mesajı ile komşularına göndermektedir. Komşularının ağırlık oranlarını öğrenen

düğümler minimum ağırlık oranını (***swr***) bulmaktadır ve ***2HOP_WRATIO(swr)*** mesajı ile komşularına göndermektedir. Tüm komşularından aldığı swr değeri kendi ağırlık oranına eşit olan düğümler kendi durumlarını (***state***) DTOR olarak değiştirmektedir ve ***CONNECT(i)*** mesajını tüm komşularına göndermektedir. ***CONNECT(i)*** mesajını alan IDLE düğümler kendi durumlarını DTEE olarak değiştirmektedir ve ***CONNECTED(i)*** mesajını tüm komşularına göndermektedir. Her düğüm tur içerisindeki işlerini bitirdiği zaman β-ağacını kullanarak turu tamamladıkları bilgisini ***OK*** mesajları ile kök düğümüne iletmektedir. Tur içerisinde en az bir düğüm DTOR olduysa kök düğümü en az bir ***OK(True)*** mesajı almaktadır ve ***START*** mesajı ile yeni turu başlatmaktadır. Kök düğüm hiç ***OK(True)*** mesajı alamaz ise ***END*** mesajı ile algoritmayı sonlandırmaktadır. ASYNSET algoritmasının ***m*** düğümü için işleyişi Algoritma 5'te verilmiştir.

**Algoritma 5:** *m* düğümü için ASYNSET algoritması

**Algoritma başladığında** *MY_ENERGY($e_m$)* mesajını tüm komşularına gönder.

***$\Gamma_m$* 'den *MY_ENERGY(e)* mesajlarını aldığında** $wr_m$ ağırlık oranını hesapla. *MY_WRATIO($wr_m$)* mesajını tüm komşularına gönder.

**Tüm DTOR olmayan komşulardan *MY_WRATIO(wr)* mesajlarını aldığında** $swr_m$ değerini bul. *2HOP_WRATIO($swr_m$)* mesajını tüm komşularına gönder.

**Tüm DTOR olmayan komşulardan *2HOP_WRATIO(swr)* mesajlarını aldığında** ∀swr = $wr_m$ ise *CONNECT(m)* mesajını tüm komşulara gönder, $state_m$=*DTOR* ve *isChanged=True*. *sync().*

***CONNECT(u)* mesajını aldığında** $state_u$=DTOR. $state_m$=IDLE ise $state_m$=DTEE ve CONNECTED(m) mesajını tüm komşularına gönder.

***CONNECTED(u)* mesajını aldığında** $state_u$=DTEE. *updateWratio().*

***START* mesajı aldığında** *isChanged* = False. *syncronized*=False. Her tur için bir kez *START* mesajını tüm komşularına gönder. $state_m$=DTOR ise sync(). $state_m$=IDLE ise *NOT_CONNECTED(m)* mesajını tüm komşularına gönder. $state_m$!=DTOR ise *updateWratio().*

***NOT_CONNECTED(u)* mesajını aldığında** *updateWratio().*

***Φ*'den *OK* mesajlarını aldığında *syncronized*=True ise** *sync().*

***END* mesajı aldığında** *END* mesajını tüm komşulara gönder. Algoritmayı sonlandır.

***updateWratio():*** Tüm IDLE komşulardan *NOT_CONNECTED* mesajı alındıysa, $wr_m$ ağırlık oranını tekrar hesapla ve *MY_WRATIO($wr_m$)* mesajını tüm komşularına gönder.

**Algoritma 5:** *m* düğümü için ASYNSET algoritmasının devamı

***sync():*** *syncronized*=True. $m=root_s \wedge \Phi$ 'den *OK* mesajlarını alındıysa ∧ (en az bir tane *OK(True)* mesajı alındıysa ∨ *isChanged*=True ) ise *START* mesajını tüm komşulara gönder. $m=root_s \wedge \Phi$ 'den OK mesajlarını alındıysa ∧ (hiç *OK(True)* mesajı alınmadıysa ∧ *isChanged*=False ) ise *END* mesajını tüm komşulara gönder ve algoritmayı sonlandır. $m!=root_s \wedge \Phi$ 'den OK mesajlarını alındıysa ∧ (en az bir tane *OK(True)* mesajı alındıysa ∨ *isChanged*=True ) ise *OK(True)* mesajını $parent_s$'a gönder. $m!=root_s \wedge \Phi$ 'den OK mesajlarını alındıysa ∧ (hiç *OK(True)* mesajı alınmadıysa ∧ *isChanged*=False ) ise *OK(False)* mesajını tüm komşulara gönder.

Algoritma 5'e göre algoritma başladığında ***m*** düğümü kendi enerjisini tüm komşularına ağırlık oranlarını hesaplayabilmeleri için ***MY_ENERGY($e_m$)*** mesajı ile göndermektedir.

Tüm komşularından ***MY_ENERGY(e)*** mesajlarını alan ***m*** düğümü IDLE komşularının ağırlıklarını ve kendi ağırlığını ***w=1/e*** şeklinde hesaplamaktadır. Ağırlıkları hesapladıktan sonra kendi ağırlık oranını $wr_m=w_m/(\sum w_u \mid u \in \Gamma_m \wedge state_u=IDLE)$ şeklinde hesaplamaktadır ve tüm komşularına ***MY_WRATIO($wr_m$)*** mesajı ile göndermektedir.

Tüm DTOR olmayan komşularından ***MY_WRATIO(wr)*** mesajlarını alan ***m*** düğümü, kendi ağırlık oranı ve komşularının ağırlık oranları içerisinden minimum ağırlık oranı olan $swr_m$ değerini bulmaktadır. Bulduğu bu değeri ***2HOP_WRATIO($swr_m$)*** mesajı ile komşularına göndermektedir.

Tüm DTOR olmayan komşularından ***2HOP_WRATIO(swr)*** mesajını alan ***m*** düğümü aldığı tüm ***swr*** değerlerini kontrol etmektedir. Tüm ***swr*** değerleri $wr_m$ değerine eşit ise ***m*** düğümü kendi durumunu DTOR olarak değiştirmektedir ve ***CONNECT(m)*** mesajını tüm komşularına göndererek bilgilendirmektedir. DTOR olan ***m*** düğümü tur içinde değişiklik yaşadığı bilgisini tutmak için ***isChanged*** değerini True olarak değiştirmektedir. Ardından ***m*** düğümü DTOR olmasına bakılmaksızın β-ağacındaki senkronizasyonu sağlamak için ***sync()*** metodunu çalıştırmaktadır.

***CONNECT(u)*** mesajını alan ***m*** düğümü ***u*** düğümünün durumunu DTOR olarak değiştirmektedir. ***m*** düğümünün durumu IDLE ise durumunu DTEE olarak değiştirmekte ve ***CONNECTED(m)*** mesajını tüm komşularına göndererek komşularını bilgilendirmektedir.

***CONNECTED(u)*** mesajını alan ***m*** düğümü ***u*** düğümünün durumunu DTEE olarak değiştirmektedir ve ağırlık oranını yeniden hesaplamak için ***updateWratio()*** metodunu çalıştırmaktadır.

***sync()*** metodunda ***m*** düğümü β-ağacındaki tüm komşularından ***OK*** mesajlarını aldıktan sonra, ***m*** düğümü $root_s$ değil ise ***isChanged*** değerine ve aldığı ***OK*** mesajlarına göre $parent_s$'e ***OK*** mesajını göndermektedir. ***isChanged*** değeri True ise veya en az bir ***OK(True)*** mesajı aldıysa $parent_s$'e ***OK(True)*** mesajını göndermektedir. Aksi takdirde ***OK(False)*** mesajını göndermektedir.

***sync()*** metodunda ***m*** düğümü β-ağacındaki tüm komşularından ***OK*** mesajlarını aldıktan sonra, ***m*** düğümü $root_s$ ise ***isChanged*** değerine ve aldığı ***OK*** mesajlarına göre ya yeni turu başlatmak için ***START*** mesajını ya da algoritmayı sonlandırmak için ***END*** mesajını tüm komşularına göndermektedir. ***isChanged*** değişkeni True ise veya en az bir ***OK(True)*** mesajı aldıysa ***START*** mesajı göndermektedir. Aksi takdirde ***END*** mesajını göndermektedir.

***START*** mesajını alan ***m*** düğümü her turda bir kez olmak üzere ***START*** mesajını tüm komşularına göndermektedir. ***isChanged*** değerini yeni turda henüz değişiklik olmadığı için False olarak değiştirmektedir. ***m*** düğümünün durumu DTOR ise DTOR seçim işlemlerine katılmayacağı için ***sync()*** metodunu çalıştırmaktadır. ***m*** düğümünün durumu IDLE ise bir önceki turda DTOR ya da DTEE olmadığını belirtmek için tüm komşularına ***NOT_CONNECTED*** mesajını göndermektedir. Ağırlık oranının tekrar hesaplanması için ***updateWratio()*** metodunu çalıştırmaktadır.

***NOT_CONNECTED*** mesajını alan ***m*** düğümü ağırlık oranının yeniden hesaplanması için ***updateWratio()*** metodunu çalıştırmaktadır.

***updateWratio()*** metodunda ***m*** düğümü tüm IDLE komşularından ***NOT_CONNECTED*** mesajlarını aldıktan sonra $wr_m$ değerini komşularının yeni durumuna göre tekrar hesaplamaktadır. Ardından ***MY_WRATIO*($wr_m$)** mesajı ile hesapladığı $wr_m$ ağırlık oranını tüm komşularına göndermektedir.

***END*** mesajını alan ***m*** düğümü komşularının algoritmanın sonladığından haberdar olmaları için ***END*** mesajını tüm komşularına göndermektedir ve algoritmayı sonlandırmaktadır.

### 6.2.2 ASYNTREE Algoritması

Bu algoritma ASYNSET algoritması tarafından bulunan düğümleri bağlı hale getirmektedir. Algoritmanın başlangıcında her düğüm komşularını (***Γ***), ağırlıklarını (***w***), β-ağacındaki çocuklarını (***Φ***) ve ebeveynini ($parent_s$) bilmektedir. DTOR düğüm komşu DTOR düğümleri ile ağaç oluşturmak için Gallager ve diğ. (1983)'nin önerdiği GHS algoritmasını çalıştırır. GHS algoritmasının sonucunda her DTOR düğümü bir ağaca bağlı olmaktadır ve bu ağaçların ID'lerini ***treeID*** değişkeninde tutmaktadır. Her ağacın ID'si o ağacın kök düğümünün ID'sine eşittir. Her DTOR düğümü içinde bulunduğu DTOR-ağacındaki kök düğümü ($root_d$), ebeveynini ($parent_d$) ve çocuklarını bilmektedir.

β-ağacının kök düğümü ***START*** mesajı ile algoritmayı başlatmaktadır. DTOR düğümler ***treeID***'lerini DTEE komşularına ***TREE_INFO(treeID)*** mesajı ile göndermektedir. DTOR komşularından ***TREE_INFO(treeID)*** mesajlarını alan DTEE düğümler komşusu oldukları ağaç bilgilerini $\Gamma^{tree}$ kümesinde tutmaktadır. DTEE düğümler DTEE komşularına ağırlık oranlarını hesaplayabilmeleri için $\Gamma^{tree}$ kümelerini ***DTEE_INFO*($\Gamma^{tree}$)** mesajı ile göndermektedir. DTEE komşularından ***DTEE_INFO*** mesajlarını alan DTEE düğümü ağırlık oranını hesaplayabilir hale gelmektedir. DTEE bir ***m*** düğümünün birden fazla komşu ağacı var ise $wr_m = w_m / |\Gamma^{tree}_m|$ şeklinde hesaplamaktadır. ***m*** düğümünün sadece bir komşu ağacı var ise ***m*** düğümü bir DTEE komşusuyla birleşerek ağırlık oranı hesaplamaktadır. Ağırlık oranı en küçük olacak şekilde bir DTEE komşusu seçmektedir. $Wr_m = (w_m + w_u) / |\Gamma^{tree}_m \ U \ \Gamma^{tree}_u|$ şeklinde hesaplanmaktadır. Hesapladıkları ***wr*** ağırlık oranlarını

***MY_WRATIO(wr)*** mesajı ile DTOR komşularına göndermektedir. DTEE komşularından ve DTOR-ağacındaki çocuklarından ***MY_WRATIO(wr)*** mesajlarını alan DTOR düğümler aldıkları en küçük ***wr*** ağırlık oranını bulmaktadır, bu oranı ***swr*** değişkeninde tutmaktadır ve ***swr*** değerini ebeveynlerine göndererek DTOR-ağaçlarındaki root düğüme ulaştırmaktadır. DTOR ağaçlarındaki root düğümleri en küçük ***wr*** ağırlık oranını bulmaktadır ve ***swr*** ağırlık oranına sahip olan düğümün ID'sini (***swrID***) ***CONNECT(swrID)*** mesajı ile DTEE komşularına göndermektedir.

DTEE bir düğüm tüm DTOR komşularından aldığı ***swrID*** değeri kendi ID'sine eşit ise tüm DTOR düğümler tarafından seçildiğini β-ağacındaki ebeveynlerine göndererek root düğüme kadar iletmektedir. β-ağacındaki root düğümü her turda en küçük ağırlık oranına sahip olan DTEE düğümünü seçmektedir ve seçtiği düğümün ID'sini ***TREE_MRG(swrID)*** mesajı ile tüm düğümlere bildirmektedir. ***TREE_MRG(swrID)*** mesajını alan düğümler seçilen düğüme göre ağaçlarını güncelleme işlemini gerçekleştirmektedir ve yeni tura başlamaktadır. β-ağacındaki root düğümünün bulduğu ***swr*** değeri sonsuz olduğunda ***END*** mesajını göndererek algoritmayı sonlandırmaktadır. Algoritma 6'da ***m*** düğümü için ASYNTREE algoritmasının işleyişi verilmiştir.

**Algoritma 6:** *m* düğümü için ASYNTREE algoritması

**Algoritma başlamadan** $state_m$ = DTOR ise GHS algoritmasını çalıştır.

***START* mesajı aldığında** $state_m$=DTOR ise *TREE_INFO(treeID)* mesajını tüm DTEE komşularına gönder.

**Tüm DTOR komşulardan *TREE_INFO(treeID)* mesajlarını aldığında** $\Gamma^{tree}{}_m$ kümesini oluştur. *DTEE_INFO(*$\Gamma^{tree}{}_m$*)* mesajını tüm DTEE komşularına gönder.

**Tüm DTEE komşulardan *DTEE_INFO(*$\Gamma^{tree}$*)* mesajlarını aldığında** $wr_m$ ağırlık oranını hesapla. *MY_WRATIO(*$wr_m$*)* mesajını tüm DTOR komşularına gönder.

**Tüm DTEE komşulardan ve DTOR-ağacındaki çocuklardan *MY_WRATIO(wr)* mesajlarını aldığında** *swr* değerini bul. $m=root_d \wedge swr=\infty$ ise *NOT_CONNECT* mesajını tüm komşularına gönder. $m=root_d \wedge swr!=\infty$ ise *CONNECT(swrID)* mesajını tüm komşularına gönder. $m!=root_d$ ise *MY_WRATIO(swr)* mesajını $parent_d$'ye gönder.

***NOT_CONNECT* mesajını aldığında** $state_m$=DTOR ise *NOT_CONNECT* mesajını her turda bir kez tüm komşularına gönder ve *sync(). statem*=DTEE ise *sync().*

***CONNECT(swrID)* mesajını aldığında $state_m$=DTOR** ise *CONNECT(swrID)* mesajını her turda bir kez tüm komşularına gönder. *sync()*

**Tüm DTOR komşulardan *CONNECT(swrID)* mesajlarını aldığında $state_m$=DTEE ise** ∀swrID = m ise *isChoosed*=True ve *sync().* ∀swr !=m ise *sync().*

***sync():*** *s*yncronized=True. *Φ*'den *CONNECT_REQ(wr, wrID)* mesajları alındıysa, *swr* hesapla. $m=root_s \wedge swr = \infty$ ise *END* mesajını tüm komşularına gönder ve algoritmayı sonlandır. $m=root_s \wedge swr != \infty$ ise *TREE_MRG(swrID)* mesajını tüm komşularına gönder. $m!=root_s$ ise *CONNECT_REQ(swr, swrID)* mesajını $parent_s$'ye gönder.

**Algoritma 6:** *m* düğümü için ASYNTREE algoritması devamı

***Φ*'den *CONNECT_REQ(wr, wrID)* mesajları aldığında *syncronized*=True ise** *sync()*.

***END* mesajı aldığında** *END* mesajını tüm komşularına gönder. Algoritmayı sonlandır.

***TREE_MRG(swrID)* mesajını aldığında** ağaç bilgilerini güncelle. *TREE_MRG(swrID)* mesajını her tur için bir kez tüm komşularına gönder. *syncronized*=False. *isChoosed*=False. *swrID*=m ise $state_m$=DTOR. $state_m$=DTOR ise *TREE_INFO(treeID)* mesajını tüm DTEE komşularına gönder.

Algoritma 6'ya göre algoritma başlamadan önce, birbirlerine komşu olan DTOR düğümler başlangıç DTOR-ağaçlarını oluşturmak için GHS algoritmasını çalıştırmaktadır.

GHS algoritması ile başlangıç ağaçları oluşturulduktan sonra β-ağacı üzerinden ***START*** mesajları gönderilerek algoritma başlatılır. ***START*** mesajını alan DTOR ***m*** düğümü ***treeID***'sini ***TREE_INFO(treeID)*** mesajı ile DTEE komşularına göndermektedir.

Tüm DTOR komşularından ***TREE_INFO(treeID)*** mesajlarını alan bir DTEE ***m*** düğümü $\Gamma^{tree}_m$ kümesini oluşturmaktadır. Ardından oluşturduğu $\Gamma^{tree}_m$ kümesini $DTEE\_INFO(\Gamma^{tree}_m)$ mesajı ile DTEE komşularına göndermektedir.

Tüm DTEE komşularından $DTEE\_INFO(\Gamma^{tree})$ mesajlarını alan bir DTEE ***m*** düğümü aldığı mesajlara göre ağırlık oranını hesaplamaktadır. ***m*** düğümü 1'den fazla ağaca komşu ise ağırlık oranını $wr_m = w_m / |\Gamma^{tree}_m|$ şeklinde hesaplamaktadır. ***m*** düğümü sadece bir ağaca komşu ise DTEE bir komşusu ile birleşerek ağırlık oranını $wr_m = (w_m + w_u) / ( |\Gamma^{tree}_m \ U \ \Gamma^{tree}_u| )$ şeklinde hesaplamaktadır. ***m*** düğümü ağırlık oranını en küçük yapan ***u*** düğümü ile birleşmektedir. Ardından, hesaplanan $wr_m$ ağırlık oranını $MY\_WRATIO(wr_m)$ mesajı ile DTOR komşularına göndermektedir.

Tüm DTEE komşularından ve DTOR-ağacındaki çocuklarından ***MY_WRATIO(wr)*** mesajlarını alan DTOR ***m*** düğümü aldığı tüm ***wr*** ağırlık oranlarını içerisinden minimum olanını (***swr***) bulmaktadır ve ***swr*** değerine sahip düğümün ID'sini ***swID*** olarak kaydetmektedir. ***m*** düğümü $root_d$ değil ise bulduğu ***swr*** ağırlık oranını ***MY_WRATIO(swr)*** mesajı ile $parent_d$'ye göndermektedir. ***m*** düğümü $root_d$ ise ***swr*** değerine göre ***NOT_CONNECT*** veya ***CONNECT(swrID)*** mesajlarını göndermektedir. ***swr*** değeri sonsuz ise ***NOT_CONNECT*** mesajını, aksi takdirde ***swr*** değerine sahip olan düğümün ID'sini ***CONNECT(swrID)*** mesajı ile tüm komşularına göndermektedir.

***NOT_CONNECT*** veya ***CONNECT(swrID)*** mesajlarını alan DTOR ***m*** düğümü her turda bir kez aldıkları mesajları tüm komşularına göndermektedir ve bu turdaki işleri bittiği için ***sync()*** metodunu çalıştırmaktadır.

***NOT_CONNECT*** mesajını alan DTEE ***m*** düğümü komşu ağaçlar tarafından seçilemediğini anlamaktadır ve bu turdaki işleri bittiği için ***sync()*** metodunu çalıştırmaktadır.

Tüm DTOR komşularından ***CONNECT(swrID)*** mesajını alan DTEE ***m*** düğümü aldığı ***swrID*** değerlerinin hepsi ***m***'e eşit ise tüm komşu ağaçları tarafından seçildiğini anlamaktadır ve ***isChosen*** değerini True olarak değiştirmektedir. Aksi takdirde seçilmediğini anlamaktadır. ***m*** düğümü seçilse de seçilmese de bu turdaki işlerini bitirdiği için ***sync()*** metodunu çalıştırmaktadır.

***sync()*** metodunda ***m*** düğümü β-ağacındaki tüm çocuklarından ***CONNECT_REQ(wr, wrID)*** mesajlarını aldıktan sonra aldığı ***wr*** değerleri arasından minimumunu (***swr***) bulmaktadır. ***m*** düğümünün β-ağacında hiç çocuğu yok ise ***CONNECT_REQ*** mesajlarını beklememektedir. ***m*** düğümünün ***isChosen*** değeri True ise bulunan ***swr*** değeri ***m*** düğümünün ağırlık oranı olan $wr_m$ ile karşılaştırılmaktadır ve küçük olan ***swr*** olarak seçilmektedir. Ardından ***m*** düğümü $root_s$ değil ise bulunan ***swr*** değeri ve ***swr*** değerine sahip olan düğümün ID'si (***swrID***) ***CONNECT_REQ(swr, swrID)*** mesajı ile $parent_s$'e gönderilmektedir. ***m*** düğümü $root_s$ değil ise, ***swr*** değerine göre ya yeni tura

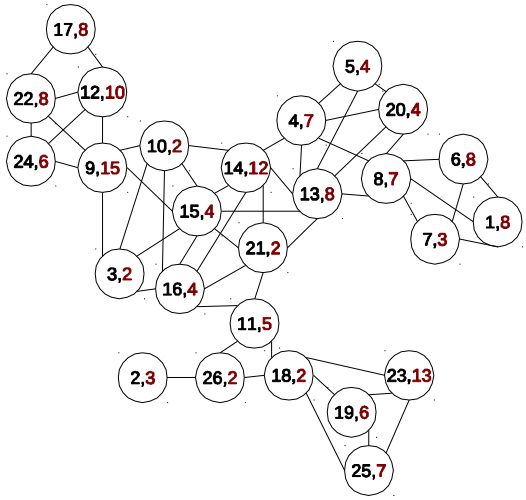
başlanmaktadır ya da algoritma sonlanmaktadır. ***swr*** değeri sonsuz ise çizgedeki tüm ağaçlar tek bir ağaç haline gelmektedir ve ***m*** düğümü ***END*** mesajını tüm komşularına göndererek algoritmayı sonlandırmaktadır. ***swr*** değeri sonsuz değil ise ***m*** düğümü ***swr*** değerine sahip olan düğümün ağaçları birleştirmesi ve ardından yeni turun başlaması için ***TREE_MRG(swrID)*** mesajını göndermektedir.

***TREE_MRG(swrID)*** mesajını alan ***m*** düğümü her turda bir kez aldığı mesajı tüm komşularına göndermektedir. ***m*** düğümünün durumu DTOR ise aldığı ***swrID*** değerine göre ağaç bilgilerini güncellemektedir. Güncelleme işlemi bittikten sonra birleşen ağaçlarının yeni ***treeID***'si ***swrID*** olmaktadır ve ***TREE_INFO(treeID)*** mesajını tüm DTEE komşularına göndermektedir. ***m*** düğümünün durumu DTEE ise ve $\boldsymbol{swrID} = \boldsymbol{m}$ ise ***m*** düğümü durumunu DTOR olarak değiştirerek birleşen ağacın kök düğümü olmaktadır ve ***TREE_INFO(treeID)*** mesajını tüm DTEE komşularına göndermektedir.

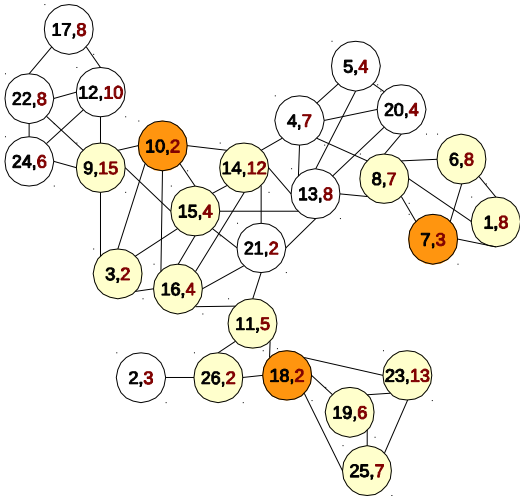
***END*** mesajını alan ***m*** düğümü algoritmanın sonlandığını komşularına ***END*** mesajı yollayarak iletmektedir ve algoritmayı sonlandırmaktadır.

### 6.2.3 ASYNSET ve ASYNTREE Algoritmalarının Örnek Üzerinde Gösterimi

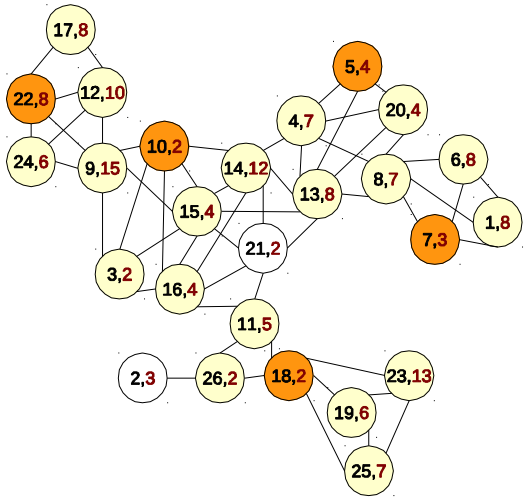
Şekil 6.2'de ASYNSET ve ASYNTREE algoritmalarının Şekil 6.1'deki çizge üzerindeki adımları verilmiştir. Şekilde beyaz ile gösterilen düğümler IDLE, turuncu ile gösterilenler DTOR ve sarı ile gösterilenler DTEE düğümlerdir. Düğümler üzerinde siyah renkte ID'leri, kırmızı renkte ağırlıkları verilmiştir.



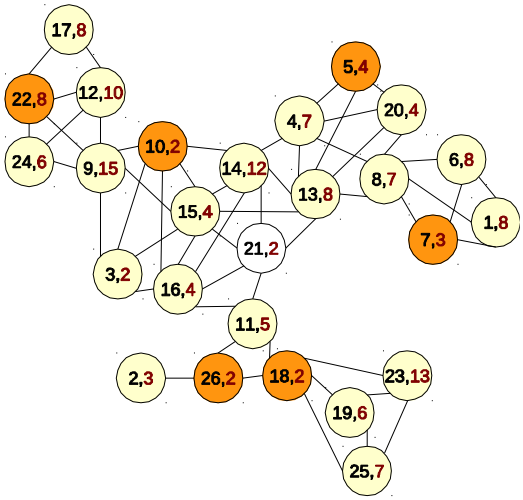

(a) Başlangıç durumu



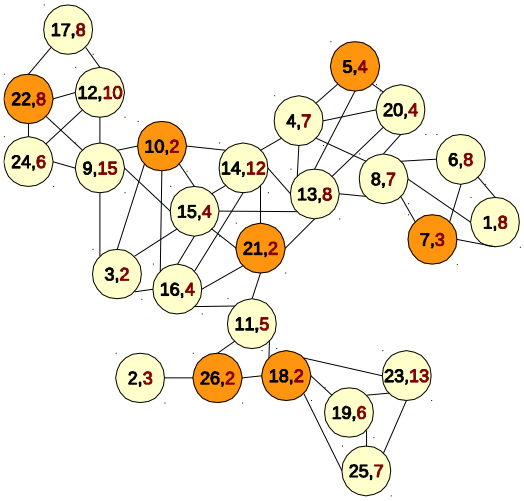

(b) Birinci tur sonunda çizgenin durumu



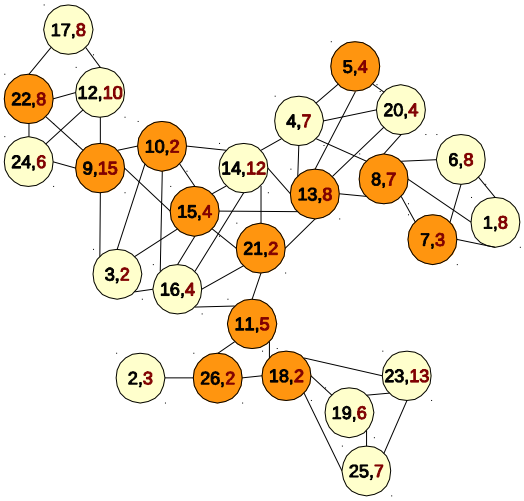

(c) İkinci tur sonunda çizgenin durumu




(d) Üçüncü tur sonunda çizgenin durumu




(e) Dördüncü tur sonunda çizgenin durumu




(d) ASYNTREE sonunda çizgenin durumu

Şekil 6.2: ASYNSET ve ASYNTREE algoritmalarının örnek işleyişi.

Düğümlerin ağırlıkları verildiği için enerji seviyelerini gönderme işlemleri yapılmış varsayılacaktır. Tüm komşularından enerji seviyelerini alan düğümler ağırlık oranlarını hesaplamaktadır ve bu ağırlık oranlarını ***MY_WRATIO(wr)*** mesajı ile tüm komşularına göndermektedir. Örneğin Şekil 6.2'de çizgenin başlangıç durumu verilmiştir ve tüm düğümler IDLE durumdadır. ***10*** numaralı düğümün ağırlık oranı ***10*** numaralı düğümün ağırlığının, ***10*** numaralı düğümün IDLE komşuları olan ***8, 9, 10***(kendisi de IDLE olduğu için), ***14, 15*** ve ***16*** numaralı düğümlerin ağırlıklarının toplamına eşittir. Dolayısıyla $wr_{10}=2/(2+15+2+4+4+12)$ denkleminden $wr_{10} = 0.051$ olarak hesaplanmaktadır ve ***10*** numaralı düğüm ***MY_WRATIO(0.051)*** mesajını tüm komşularına göndermektedir. Bazı düğümlerin ağırlık oranları aşağıdaki gibi olmaktadır.

$wr_{15} = 4 / (4+2+2+4+15+12+2+8) = 0.081$

$wr_{2} = 3 / (3+2) = 0.66$

$wr_{7} = 3 / 3+7+8+8 = 0.11$

Tüm DTOR olmayan (DTEE ya da IDLE olan) komşularından ***MY_WRATIO*** mesajını alan DTOR olmayan düğümler komşularının ve kendilerinin ağırlık oranları içerisinde en küçük ağırlık oranını bulmaktadırlar ve buldukları en küçük ağırlık oranını ***2HOP_WRATIO(swr)*** mesajı ile tüm komşularına göndermektedir. Örneğin ***8, 9, 10, 14, 15*** ve ***16*** numaralı düğümlerin buldukları en küçük ağırlık oranı ***10*** numaralı düğümün ağırlık oranıdır ve bu düğümler ***2HOP_WRATIO(0.051)*** mesajını tüm komşularına göndermektedir.

Tüm DTOR olmayan komşularından ***2HOP_WRATIO*** mesajlarını alan DTOR olmayan düğümler, aldıkları her ***2HOP_WRATIO*** mesajındaki minimum ağırlık oranı kendi ağırlık oranına eşit ise DTOR olmakta ve ***CONNECT*** mesajını tüm komşularına göndermektedir. Örneğin ***10*** numaralı düğüm DTOR olmayan tüm komşularından ***2HOP_WRATIO(0.051)*** mesajını almaktadır ve ***10*** numaralı düğümün ağırlık oranı da ***0.051***'dir. Dolayısıyla ***10*** numaralı düğüm DTOR olmaktadır. ***isChanged*** değerini True olarak değiştirmektedir ve ***CONNECT(10)*** mesajını tüm komşularına göndermektedir. ***10*** numaralı düğüm gibi ilk turda ***7*** ve ***18*** numaralı düğümler de DTOR olmaktadır. ***10, 7*** ve ***18*** numaralı düğümler ***CONNECT***

mesajlarını gönderdikten sonra ***sync()*** metodunu çalıştırmaktadır ve ***isChanged*** değişkenleri True olduğu için ***OK(True)*** mesajını ebeveynlerine göndermektedir.

Tüm DTOR olmayan komşularından ***2HOP_WRATIO*** mesajlarını alıp DTOR olamayan düğümler de ***sync()*** metodunu çalıştırmaktadır.

***10*** numaralı düğümden ***CONNECT(10)*** mesajını alan ***8, 9, 14, 15*** ve ***16*** numaralı düğümler IDLE durumda oldukları için DTEE olmaktadır ve ***CONNECTED*** mesajlarını tüm komşularına göndermektedir. ***10*** numaralı düğümün komşuları gibi ***CONNECT*** mesajını alan ***1, 6, 8, 11, 19, 23, 25*** ve ***26*** numaralı düğümler de IDLE durumunda oldukları için DTEE olmaktadır.

Tüm düğümler ***sync()*** metotlarını çalıştırdıklarında β-ağacının kök düğümü tur bittiğinde tur içerisinde yeni DTOR düğümlerin seçileceği için yeni turu başlatmak üzere ***START*** mesajını tüm çizgeye göndermektedir.

START mesajını alan IDLE durumda olan ***2, 4, 5, 12, 13, 17, 20, 21, 22*** ve ***24*** numaralı düğümler bir önceki turda DTOR ya da DTEE olamadıkları için ***NOT_CONNECT*** mesajlarını tüm komşularına göndermektedir.

Tüm IDLE komşularından ***NOT_CONNECT*** mesajı alan DTOR olmayan düğümler ağırlık oranlarını yeniden hesaplamaktadır ve ***MY_WRATIO*** mesajlarını tüm komşularına göndermektedir. Örneğin ***22*** numaralı düğümün ağırlık oranı kendi ağırlığının IDLE komşuları olan ***12, 17, 22*** ve ***24*** numaralı düğümlerin ağırlıkları toplamına oranıdır. Dolayısıyla $wr_{22}=8/(8+8+10+6)=0.25$ olarak hesaplanmaktadır.

Tüm DTOR olmayan komşularından ***MY_WRATIO*** mesajlarını alan ***12, 17, 22*** ve ***24*** numaralı düğümler buldukları minimum ağırlık 0.25 olduğu için ***2HOP_WRATIO(0.25)*** mesajını, ***4, 5, 13 ve 20*** numaralı düğümler ise ***2HOP_WRATIO(0.16)*** mesajını tüm komşularına göndermektedir.

Tüm DTOR olmayan komşularından ***2HOP_WRATIO*** mesajları ile kendi ağırlık oranlarını alan ***5*** ve ***22*** numaralı düğümler DTOR olmaktadır ve ***CONNECT(5), CONNECT(22)*** mesajlarını göndermektedir.

***CONNECT(5)*** veya ***CONNECT(22)*** mesajlarını alan ***4, 12, 13, 17, 20*** ve ***24*** numaralı düğümler IDLE durumda oldukları için DTEE olmaktadır.

Tüm düğümler ***sync()*** metodunu çalıştırdıktan sonra ikinci tur bitmektedir ve tur sonunda çizgenin durumu Şekil 6.2(c)'de verilmiştir. Tur içerisinde DTOR seçildiği için üçüncü tur başlatılmaktadır.

Üçüncü turda algoritma aynı şekilde çalışarak ***26*** numaralı düğümü DTOR, ***2*** numaralı düğümü DTEE yapmaktadır. Üçüncü tur sonunda çizgenin durumu Şekil 6.2(d)'de verilmiştir. Dördüncü turda ise ***21*** numaralı düğümü DTOR yapmaktadır ve tur sonunda çizgenin durumu Şekil 6.2(e)'de verilmiştir. Beşinci turda herhangi bir düğüm DTOR olmadığı için ASYNSET algoritması sonlanmakta ve GHS algoritması başlamaktadır.

GHS algoritması sonucunda ***5, 7, 10, 21*** ve ***22*** numaralı düğümler tek düğümlü ağaçlar oluşturmaktadır ve ***treeID***'leri kendi ID'leridir. ***18*** ve ***26*** numaralı düğümler ise birleşerek bir ağaç oluşturmaktadır ve ***treeID***'leri ***26***'dır.

GHS algoritmasından sonra ***TREE_INFO*** ve ***DTEE_INFO*** mesajları gönderilmektedir. ***DTEE_INFO*** mesajlarını alan DTEE düğümler ağırlık oranlarını hesaplamaktadır ve ***MY_WRATIO*** mesajlarını DTOR komşularına göndermektedir. Örneğin ***15*** numaralı düğüm birden fazla ağaca komşu olduğu için ağırlık oranı $wr_{15}=w_{15}/|\Gamma^{tree}_{15}|$ şeklinde hesaplanmaktadır. $w_{15}=4$ ve $\Gamma^{tree}_{15}=\{10,21\}$ olduğundan $wr_{15} = 4/2 = 2$ olarak hesaplanmaktadır. ***15*** numaralı düğüm ***MY_WRATIO(2)*** mesajını tüm DTOR komşularına göndermektedir. ***8*** numaralı düğümün sadece bir komşu ağacı olduğu için bir komşusu ile birleşerek ağırlık oranını hesaplamaktadır. En küçük ağırlık oranını ***20*** numaralı düğüm ile birleşerek yakalamaktadır ve $wr_8 = (w_8+w_{20}) / |\Gamma^{tree}_8 \ U \ \Gamma^{tree}_{20}|$ olarak hesaplanmaktadır. $w_8=7$, $w_{20}=4$, $\Gamma^{tree}_8=\{7\}$ ve $\Gamma^{tree}_{20}=\{5\}$ olduğundan $wr_8=(7+4)/(1+1)=5.5$ olmaktadır.

Tüm DTEE komşularından ***MY_WRATIO*** mesajlarını alan düğümler kendi DTOR-ağaçları içerisinde ağacın komşu olduğu en küçük ağırlık oranına sahip olan düğümü bularak ***CONNECT*** mesajlarını göndermektedir. Dolayısıyla ***10*** ve ***21*** numaralı ağaçlar tüm komşularına ***CONNECT(15)*** mesajını göndermektedir. ***5*** numaralı ağaç ***CONNECT(13)***, ***7*** numaralı ağaç ***CONNECT(8)***, ***28*** numaralı ağaç

***CONNECT(9)*** ve ***26*** numaralı ağaç ise ***CONNECT(11)*** mesajlarını tüm DTEE komşularına göndermektedir.

Tüm ağaç komşularından ***CONNECT*** mesajları ile kendi ID'sini alan ***15*** numaralı düğüm β-ağacı içerisinde DTOR seçilmek için aday olmaktadır.

Tüm düğümler tur içindeki işlerini bitirdiğinde β-ağacınındaki yaprak düğümlerden başlayarak ***CONNECT_REQ*** mesajlarını β-ağacının kök düğümüne doğru en küçük ağırlık oranına sahip ve tüm komşu ağaçları tarafından seçilmiş olan düğümü bulmaktadır. Bu tur için tüm komşu ağaçları tarafından seçilen tek düğüm ***15*** numaralı düğüm olduğu için en küçük ağırlık oranına sahip düğüm ***15*** numaralı düğüm seçilmektedir ve ***TREE_MRG(15)*** mesajı tüm çizgeye gönderilmektedir.

***TREE_MRG(15)*** mesajını alan ***15*** numaralı düğüm ***10*** ve ***21*** numaralı ağaçları birbirlerine bağlayarak kendisini DTOR ve $root_d$ yapmaktadır ve yeni oluşturulan ağacın ID'si ***15*** olmaktadır. ***TREE_MRG(15)*** mesajını alan ***10*** ve ***21*** numaralı ağaçlar kendi DTOR-ağaç bilgilerini güncellemektedir. Sonuç olarak bu tur sonunda birleştirilerek oluşan ağaç kökü ***15*** numaralı düğüm olan ve kök düğümün çocuğu olan ***10*** ve ***21*** numaralı düğümlerden oluşmaktadır. Ardından ***TREE_MRG(15)*** mesajını alan tüm DTOR düğümler ***TREE_INFO*** mesajlarını göndererek ikinci tura başlamaktadır.

İkinci turda toplam ***5*** ağaç bulunmaktadır ve bunlar ***5, 7, 15, 22*** ve ***26*** numaralı ağaçlardır. ***5*** numaralı ağaç ***13*** numaralı, ***7*** numaralı ağaç ***8*** numaralı, ***15*** ve ***26*** numaralı ağaç ***11*** numaralı, ***22*** numaralı ağaç ise ***9*** numaralı düğümleri seçmektedir. Tüm komşu ağaçları tarafından seçilen en küçük ağırlığa sahip olan düğüm ***11*** numaralı düğüm olduğu için ***11*** numaralı düğüm DTOR seçilmektedir ve ***15*** numaralı ağaç ile ***26*** numaralı ağacı bağlayarak ***11*** numaralı yeni bir ağaç oluşturmaktadır.

Üçüncü turda ***13***, dördüncü turda ***8***, beşinci turda ***9*** numaralı düğümler DTOR olarak seçilmekte ve komşu ağaçlarını birleştirmektedir. Altıncı turda ise tüm DTEE düğümlerin ağırlık oranları sonsuz olduğu ve çizgedeki tüm DTOR düğümler tek bir ağaçta birleştiği için algoritma sonlanmaktadır. Sonuç olarak ***5, 7, 8, 9, 10, 11, 13,***

***15, 18, 21, 22*** ve ***26*** numaralı düğümler bağlı baskın kümeyi oluşturmaktadır. Çizgenin son durumu Şekil 6.2(f)'de verilmiştir.

## 6.3 CN-MWCDS Algoritması

CN-MWCDS algoritması WANG ve ASYNSET-ASYNTREE algoritmaları gibi minimum ağırlıklı bağlı baskın küme (MWCDS) oluşturma algoritmasıdır ve diğer algoritmalar gibi çift fazlı bir algoritmadır. Algoritmanın ilk fazında bağlı olmayan bir baskın küme oluşturulmakta ve ikinci fazında ise baskın kümedeki düğümlere yeni düğümler eklenerek baskın kümedeki düğümler birbirlerine bağlanmaktadır.

Algoritma birim disk çizgeler üzerinde çalışmakta olup birinci fazın başlayabilmesi için düğümler komşularını ($\Gamma$), 2-kenar komşularını ve komşularının ağırlıklarını öğrenmek zorundadır. 2-kenar komşuluk bilgilerini ve komşularının ağırlıklarını öğrenmek için düğümler belirli aralıklarla belirli sayıda ***HELLO*** mesajı göndermektedir. Bir $m$ düğümünün gönderdiği $HELLO_m$ mesajının içerisinde $m$ düğümünün komşuları ($\Gamma_m$) ve $m$ düğümünün ağırlığı ($w_m$) bulunmaktadır. $HELLO_u$ mesajını alan $m$ düğümü, $u$ düğümünü $\Gamma_m$ kümesine eklemektedir ve $w_u$ ile $\Gamma_u$ bilgilerini kaydetmektedir. ***HELLO*** mesajları bittikten sonra düğümler komşularını, komşularının komşularını ve komşularının ağırlıklarını öğrenmiş olmaktadır.

CN-MWCDS algoritmasının ilk fazında düğümlerin birbirleri ile bağlantı kurabilme ihtimallerini de hesaba katarak birbirlerine daha kolay bağlanabilen kritik düğümler ile bir baskın küme oluşturulmaktadır. Dolayısıyla ikinci faza daha az görev düşmektedir. Algoritmada her düğümün algoritma için aktif olup olmadıklarını tutan ***isActive*** değeri bulunmaktadır ve algoritma aktif düğümler arasında çalışmaktadır. $m$ düğümünün aktif komşuları $\Gamma^{active}_m$ kümesinde tutulmaktadır. Algoritmanın ilk fazında DTOR düğümler seçilirken ***Pwr*** değerleri kullanılmaktadır. Her turda 2-kenar uzaklıktaki aktif komşuları içerisinde en yüksek ***Pwr*** değerine sahip olan düğüm ya da düğümler DTOR olmaktadır. ***Pwr*** değerleri hesaplanırken ağırlık oranları kullanılmaktadır. Her düğümün her komşusuna özel bir ağırlık oranı bulunmaktadır ve $m$ düğümünün $u$ düğümüne özel hesapladığı ağırlık oranı $wr_{m,u}$ şeklinde gösterilmektedir. $wr_{m,u} = w_m / ( I_m * (|\Gamma_m \setminus \Gamma_u|+1))$ şeklinde

hesaplanmaktadır. $I_m$, $m$ düğümünün IDLE komşu sayısını, $w_m$ $m$ düğümünün ağırlığını göstermektedir. Ne kadar IDLE düğüme komşu ise DTOR seçildiğinde o kadar fazla düğümü kapsayacağı için $I_m$ değeri bölüm olarak eklenmiştir. DTOR düğümlerin birbirlerine bağlanma maliyetlerinin düşürülmesi için düğümler arasındaki farklı komşu sayıları da bölüm olarak eklenerek daha farklı düğümlere bağlanabilen düğümlerin seçilme olasılığı artırılmıştır. Bir $m$ düğümünün $u$ komşusuna olan ağırlık oranı, $m$ düğümünün $u$ düğümünden ne kadar faklı komşuya sahip olduğuna, kendi ağırlığına ve IDLE komşu sayısına bağlıdır. $Pwr_m$ değeri ise $m$ düğümünün kaç tane $u$ komşusu için $wr_{m,u}$ ağırlık oranının $wr_{u,m}$ ağırlık oranından küçük olduğunu tutmaktadır. $Pwr_m = |\{u \mid wr_{m,u} <= wr_{u,m} \ \wedge u \in \Gamma^{active}_m\}|$ şeklinde hesaplanmaktadır.

Algoritmanın ikinci fazında oluşturulan baskın kümeye yeni düğümler eklenerek bağlı baskın küme oluşturulmaktadır. DTOR düğümleri bağlayacak düğümler seçilirken ilk fazdakinden farklı bir ağırlık oranı kullanılmaktadır. DTEE bir $m$ düğümünün ağırlık oranı $wr_m$ şeklinde gösterilmektedir. Bir $m$ düğümüne komşu olan ağaçlar $\Gamma^{tree}_m$ kümesinde tutulmaktadır. $m$ düğümü, ağırlık oranını sadece kendisi ile veya bir komşusu ile birleşerek hesaplayabilmektedir. Sadece kendisini kullanarak $wr = w_m / C\left(\left|\Gamma^{tree}_m\right|, 2\right)$ şeklinde hesaplamaktadır. Bir $u$ komşusu ile birleşerek $wr = (w_m+w_u) / C\left(\left|\Gamma^{tree}_m \cup \Gamma^{tree}_u\right|, 2\right)$ şeklinde hesaplamaktadır. Genel olarak özetlemek gerekirse bir düğümün ağırlık oranı, işleme kattığı düğümlerin toplam ağırlığının, komşu oldukları ağaç sayısının ikili kombinasyonuna oranıdır. ASYNTREE algoritması komşu ağaç sayısına göre oranı kullanmaktadır ve buradaki fark ağaç sayısı yerine ağaç sayısının ikili kombinasyonunun kullanılmasıdır. Bunun sebebi bağlanılan ağaç sayısına verilen önemi arttırmaktır. $m$ düğümü hesapladığı ağırlık oranları içerisinden en küçük olanını kendi ağırlık oranı olarak kullanmaktadır ve aşağıdaki $wr_m = min(\{ w_m / C\left(\left|\Gamma^{tree}_m\right|, 2\right)\} \cup \{(w_m+w_u) / C\left(\left|\Gamma^{tree}_m \cup \Gamma^{tree}_u\right|, 2\right) \mid u \in \Gamma^{DTEE}_m \})$ şeklinde hesaplanmaktadır.

Her turda tüm ağaçlar komşu oldukları DTEE düğümler içerisinden en küçük ağırlık oranına sahip olan düğümü seçmektedir. Tüm komşu ağaçları tarafından varsa partnerleri ile birlikte yoksa tek başına seçilen düğümler DTOR olmakta ve komşu

ağaçlarını birleştirmektedir. Çizgede tek bir ağaç kalana kadar turlar devam etmektedir.

### 6.3.1 CN-MWCDS Algoritması Birinci Fazı

Algoritmanın başlangıcında her ***m*** düğümü enerji seviyesi ile ters orantılı olan ağırlığını ($w_m$), komşularını ($\Gamma_m$), komşularının ağırlıklarını ($w_u$), komşularının komşularını ($\Gamma_u$) ve aktif komşularını ($\Gamma^{active}_m$) bilmektedir. Özet olarak her turda aktif düğümler IDLE komşu sayıları olan ***I*** değerlerin hesaplamakta ve aktif komşularına göndermektedir. Aktif komşularından ***I*** değerlerini toplayan düğümler her komşusu için ağırlık oranlarını hesaplayarak ***Pwr*** değerlerini hesaplamaktadır. Ardından ***Pwr*** değerlerini tüm aktif komşularına göndermektedir. Tüm aktif komşularından ***Pwr*** değerlerini toplayan düğümler en büyük ***Pwr*** değerine sahip olan komşusunun ID'sini ***MaxPwrID*** mesajı ile tüm aktif komşularına göndermektedir. Tüm aktif komşularından kendi ID'sini alan düğümler DTOR komşuları ise DTEE olmaktadır. Tur sonunda DTOR olan ve DTOR olma ihtimali kalmayan düğümler pasif hale geçmektedir.

Algoritmanın ***m*** düğümü için detaylı işleyişini gösteren sonlu durum makinesi Şekil 6.3'te verilmiştir.

Diyagrama göre tüm düğümler ***IDLE*** durumunda başlamaktadır. Bu durumda herhangi bir ***u*** komşusundan $I_u$ mesajı aldığında $I_u$ değerini kaydetmektedir. Algoritma anlatılırken ***IDLE***, ***DTOR*** ve ***DTEE*** sonlu durumları ile karıştırılmaması için ***state*** değişkenlerinin alabileceği değerler Idle, Dtor ve Dtee şeklinde gösterilmiştir.

***m*** düğümü algoritma başladığında veya tüm aktif komşularından ***STATUS*** mesajlarını aldığında $I_m$ değerini $I_m=|\{ k \mid k \in \Gamma_m \wedge state_k = Idle \}|$ şeklinde hesaplamaktadır. $Pwr_m$ değerini sıfırlamaktadır. Önceden alınan her $I_u$ değeri için $updatePwr(I_u)$ metodunu çalıştırmaktadır. Ardından ***Pwr Hesaplama*** durumuna geçmektedir.

***PWR HESAPLAMA*** durumunda ***m*** düğümü herhangi bir ***u*** komşusundan $I_u$ değerini alırsa ***updatePwr*($I_u$)** metodunu çalıştırarak $Pwr_m$ değerini güncellemektedir. ***m*** düğümü $Pwr_u$ değerini alırsa $Pwr_u$ değerini kaydetmektedir.

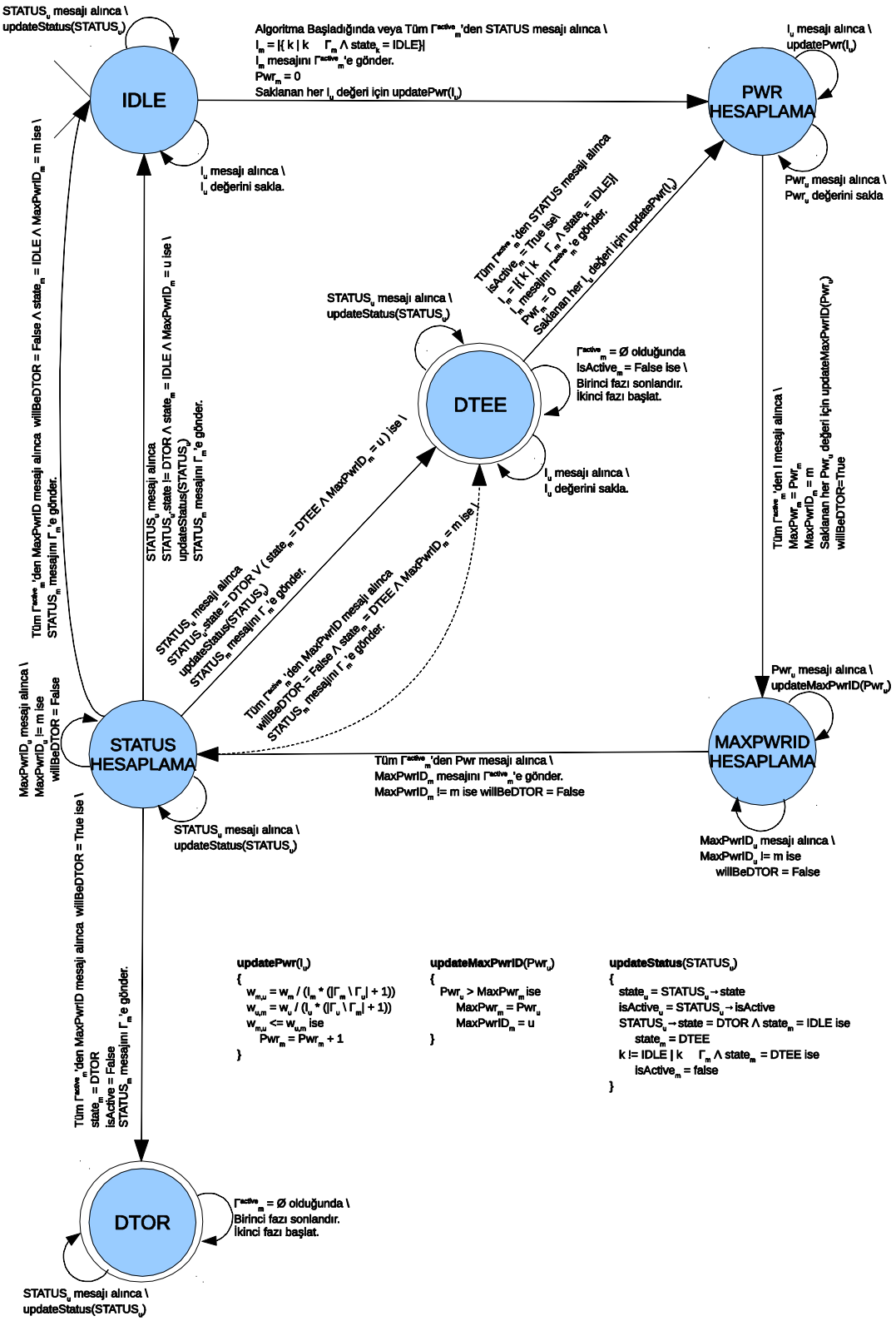
***PWR HESAPLAMA*** durumunda ***m*** düğümü tüm aktif komşularından ***I*** değerlerini aldıktan sonra $MaxPwr_m$ değerini $Pwr_m$ ile, $MaxPwrID_m$ değerini ***m*** ile ve Dtor olup olmayacağını tutan ***willBeDTOR*** değişkenini ***True*** ile başlatmaktadır. Önceden kaydedilen her $Pwr_u$ değeri için ***updateMaxPwrID*($Pwr_u$)** metodunu çalıştırmaktadır. Daha sonra ***MaxPwrID Hesaplama*** durumuna geçmektedir.

***MAXPWRID HESAPLAMA*** durumunda ***m*** düğümü herhangi bir ***u*** komşusundan $Pwr_u$ değerini alırsa ***updateMaxPwrID*($Pwr_u$)** metodunu çalıştırarak $MaxPwrID_m$ değerini güncellemektedir. $MaxPwrID_u$ değerini alırsa ve $MaxPwrID_u$ değeri ***m***'e eşit değil ise Dtor olamadığını anlamaktadır ve ***willBeDTOR*** değerini False ile değiştirmektedir.

***MAXPWRID HESAPLAMA*** durumunda ***m*** düğümü tüm aktif komşularından ***Pwr*** değerlerini aldığında $MaxPwrID_m$ değerini tüm aktif komşularına göndermektedir. $MaxPwrID_m$ değeri ***m***'e eşit değil ise Dtor olamadığını anlamaktadır ve $willBeDTOR_m$ değerini False yapmaktadır. Ardından ***Status Hesaplama*** durumuna geçmektedir.

***STATUS HESAPLAMA*** durumunda ***m*** düğümü herhangi bir ***u*** komşusundan $MaxPwrID_u$ değerini alırsa ve $MaxPwrID_u$ değeri ***m***'e eşit değil ise Dtor olamadığını anlamaktadır ve $willBeDTOR_m$ değerini False ile değiştirmektedir. $MaxPwrID_m$ hariç herhangi bir komşusundan $STATUS_u$ mesajı alırsa ***updateStatus*($STATUS_u$)** metodunu çalıştırarak ***u*** düğümünün durum ve aktiflik bilgilerini güncellemektedir.

***STATUS HESAPLAMA*** durumunda ***m*** düğümü tüm aktif komşularından ***MaxPwrID*** değerlerini aldığında $willBeDTOR_m$ değeri True ise ***m*** düğümü Dtor olmaktadır ve $state_m$ değerini Dtor yapmaktadır. Dtor olduğu için $isActive_m$ değerini False yaparak pasif olmakta ve komşularından ***STATUS*** mesajlarını beklemektedir. $state_m$ ve $isActive_m$ değerlerini $STATUS_m$ mesajı ile tüm komşularına göndermektedir. Daha sonra ***DTOR*** durumuna geçmektedir.




Şekil 6.3: CN-MWCDS algoritmasının birinci fazının m düğümü için sonlu durum diyagramı.

***STATUS HESAPLAMA*** durumunda ***m*** düğümü tüm aktif komşularından ***MaxPwrID*** değerlerini aldığında $willBeDTOR_m$ değeri False ve $MaxPwrID_m$ değeri ***m*** ise $STATUS_m$ mesajını tüm komşularına göndermektedir. Ardından $state_m$ değeri Idle ise ***IDLE*** durumuna, Dtee ise ***DTEE*** durumuna geçmektedir.

***STATUS HESAPLAMA*** durumunda ***m*** düğümü $MaxPwrID_m$ komşusundan $STATUS_u$ mesajını aldığında ***updateStatus(***$STATUS_u$***)*** metodunu çalıştırmaktadır ve $STATUS_m$ mesajını tüm komşularına göndermektedir. Daha sonra $state_m$ değeri Idle ise ***IDLE*** durumuna, Dtee ise ***DTEE*** durumuna geçmektedir.

***DTEE*** durumunda ***m*** düğümü $MaxPwrID_m$ hariç herhangi bir ***u*** komşusundan $STATUS_u$ mesajı alırsa ***updateStatus(***$STATUS_u$***)*** metodunu çalıştırmaktadır. ***m*** düğümü $I_u$ değerini alırsa $I_u$ değerini kaydetmektedir.

***DTEE*** durumunda ***m*** düğümü tüm aktif komşularından ***STATUS*** mesajlarını aldığında $isActive_m$ True ise $I_m$ değerini hesaplamaktadır ve tüm aktif komşularına göndermektedir. $Pwr_m$ değerini sıfırlamaktadır ve önceden aldığı her $I_u$ değeri için ***updatePwr(***$I_u$***)*** metodunu çalıştırmaktadır. Daha sonra ***Pwr Hesaplama*** durumuna geçmektedir.

***DTEE*** durumunda ***m*** düğümünün aktif komşusu kalmadığında kendisi de pasif ise algoritmayı sonlandırmaktadır. Pasif olma durumu ***updateStatus*** metodunda anlatılmaktadır.

***DTOR*** durumunda ***m*** düğümü herhangi bir ***u*** komşusundan $STATUS_u$ mesajı alırsa ***updateStatus(***$STATUS_u$***)*** metodunu çalıştırmaktadır. Aktif komşusu kalmadığında algoritmayı sonlandırmaktadır.

***updatePwr(***$I_u$***)*** metodunda ***m*** düğümü $wr_{m,u}$ ve $wr_{u,m}$ ağırlık oranlarını hesaplamaktadır. $wr_{m,u}$ ağırlık oranı $wr_{u,m}$ ağırlık oranından küçük veya eşit ise $Pwr_m$ değerini bir artırmaktadır.

***updateMaxPwrID(***$Pwr_u$***)*** metodunda ***m*** düğümü $MaxPwr_m$ değeri ile $Pwr_u$ değerini karşılaştırmaktadır. $Pwr_u$ değeri büyük ise $MaxPwr_m$ değerini $Pwr_u$ ile $MaxPwrID_m$ değerini ***u*** ile değiştirmektedir. Eşit oldukları durumda ise ID'si küçük olan seçilmektedir.

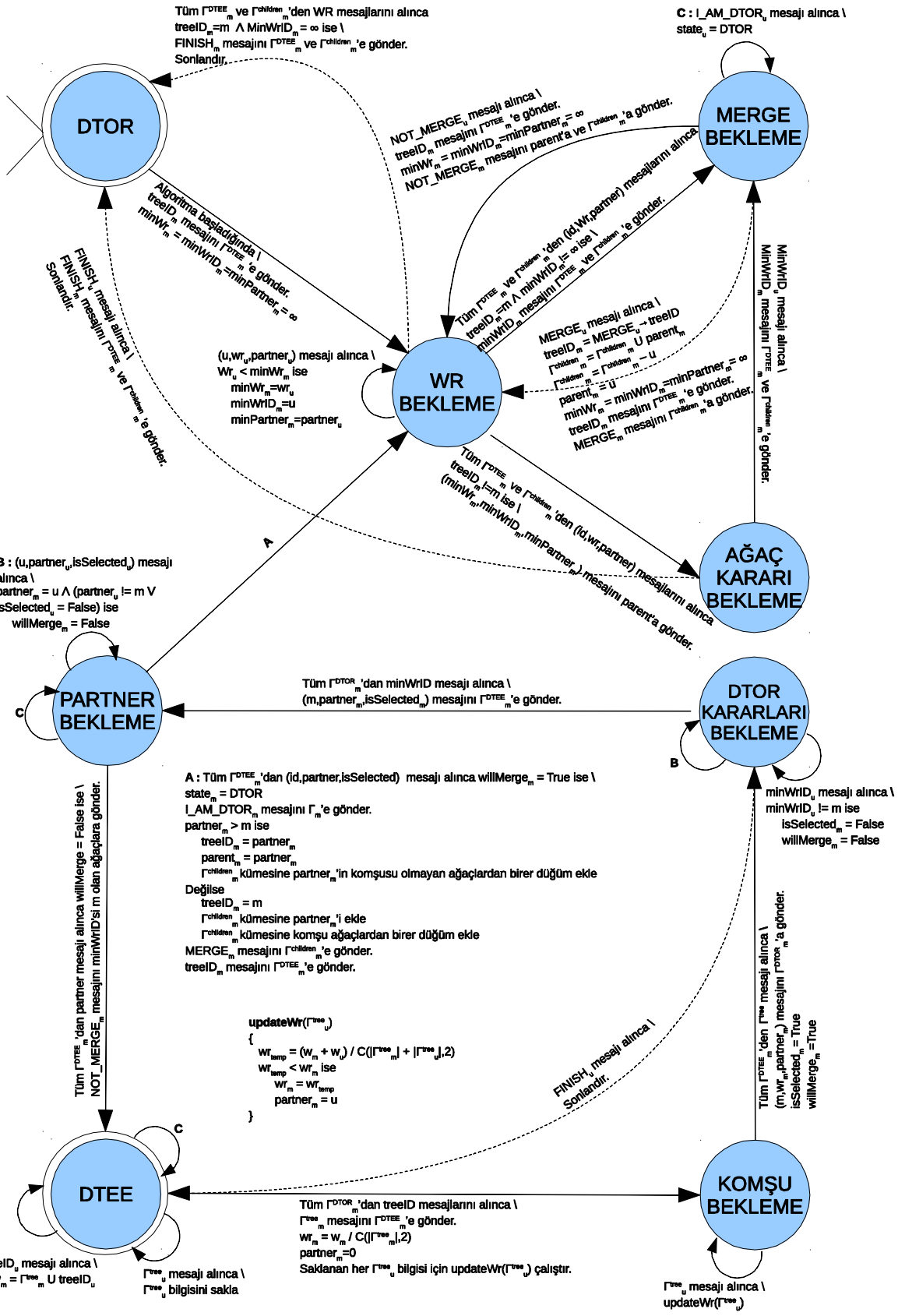
***updateStatus(STATUS$_u$)*** metodunda ise ***m*** düğümü ***STATUS$_u$*** mesajına göre hem ***u*** düğümünün hem de kendisinin ***state*** ve ***isActive*** değerlerini güncellemektedir. ***state$_u$*** değerini ***STATUS$_u$→state***, ***isActive$_u$*** değerini ***STATUS$_u$→isActive*** yapmaktadır. ***state$_m$*** değeri Idle ve ***state$_u$*** değeri Dtor ise ***state$_m$*** değerini Dtee olarak değiştirmektedir. ***m*** düğümünün Idle komşusu kalmadıysa ve ***state$_m$*** değeri Idle değil ise ***isActive$_m$*** değerini False yaparak pasif olmaktadır.

### 6.3.2 CN-MWCDS Algoritmasının İkinci Fazı

Dtor düğümler kendi aralarında Gallager ve diğ. (1983)'nin önerdiği GHS algoritmasını çalıştırmaktadır. Bu algoritmanın sonucunda birbirine komşu olan Dtor düğümler birleşerek başlangıç ağaçlarını oluşturmaktadır. CN-MWCDS algoritmasının ikinci fazının başlangıcında tüm düğümler komşularını ($\Gamma_m$) ve komşularının ***state*** değerlerini bilmektedir. Dtor düğümler içinde bulundukları ağacın kök düğümünün ID'sini ***treeID*** değişkeninde tutmaktadır. Ağaç içerisindeki ebeveynlerini ***parent*** değişkeninde ve çocuklarını $\Gamma^{children}$ kümesinde tutmaktadır. Dtee düğümler ise kendi ağırlıklarını ve komşularının ağırlıklarını bilmektedir.

Algoritma özet olarak her turda Dtor düğümler ***treeID*** değerlerini Dtee komşularına göndermektedir. ***treeID*** değerlerini toplayan Dtee düğümler ***treeID*** değerlerini tuttukları $\Gamma^{tree}$ kümelerini Dtee komşularına göndermektedir. Dtee komşularından $\Gamma^{tree}$ kümelerini toplayan Dtee düğümler ağırlık oranlarını (***wr***) hesaplamaktadır ve komşu ağaçlarına göndermektedir. Dtee komşularından ***wr*** değerlerini toplayan ağaçlar en küçük ***wr*** değerine sahip olan düğümün ID'sini (***minWrID***) tüm Dtee komşularına göndermektedir. Komşu ağaçlarından ***minWrID*** değerlerini toplayan Dtee düğüm ya da düğümler tüm ağaç komşularından aldıkları ***minWrID*** değerleri kendi ID'lerine eşit ise komşu ağaçlarına ***MERGE*** mesajını göndererek komşu ağaçlarını tek bir ağaçta bağlamaktadır ve yeni oluşan ağacın kök düğümü ***MERGE*** mesajını gönderen Dtee düğüm olmaktadır. Çizgede tek bir ağaç kalana kadar turlar devam etmektedir. Bu işlemler, çizgede bir ağaç kaldığında bağlı baskın küme oluşmaktadır ve algoritma sonlanmaktadır. Bir ***m*** düğümü için algoritmanın detaylı sonlu durum makinesi Şekil 6.4'te verilmiştir.

Algoritmada ***state*** değeri Dtor olan düğümler ***DTOR, WR BEKLEME, MERGE BEKLEME*** ve ***AĞAÇ KARARI BEKLEME*** durumlarında, Dtee olan düğümler ***DTEE, KOMŞU BEKLEME, DTOR KARARLARI BEKLEME*** ve ***PARTNER BEKLEME*** durumlarında olabilirler.




Şekil 6.4: CN-MWCDS algoritmasının ikinci fazının m düğümü için sonlu durum diyagramı

***DTOR*** durumunda ***m*** düğümü algoritma başladığında $treeID_m$ değerini Dtee komşularına göndermektedir. En küçük ağırlık oranını tutan $minWr_m$, en küçük ağırlık oranına sahip düğümün ID'sini tutan $minWrID_m$ ve en küçük ağırlık oranına sahip düğümün partner bilgisini tutan $minPartner_m$ değerlerini sonsuza eşitlemektedir ve ***WR BEKLEME*** durumuna geçmektedir.

***DTEE*** durumunda ***m*** düğümü herhangi bir ***u*** komşusundan $treeID_u$ değerini alırsa $\Gamma^{tree}_m$ kümesini $\Gamma^{tree}_m = \Gamma^{tree}_m \ U \ \{treeID_u\}$ şeklinde güncellemektedir. $\Gamma^{tree}_u$ kümesini alırsa $\Gamma^{tree}_u$ değerini kaydetmektedir. $I\_AM\_DTOR_u$ mesajını aldığında ise $state_u$ değerini Dtor olarak değiştirmektedir.

***DTEE*** durumunda ***m*** düğümü tüm Dtor komşularından ***treeID*** değerlerini toplayan ***m*** düğümü $\Gamma^{tree}_m$ kümesini Dtee komşularına göndermektedir. $wr_m$ değerini $w_m/C\binom{|\Gamma^{tree}_m|}{2}$ , partner bilgisinin tutulduğu $partner_m$ değerini 0 olarak değiştirmektedir. Önceden alınan her $\Gamma^{tree}_u$ kümesi için $updateWr(\Gamma^{tree}_u)$ metodunu çalıştırmaktadır. Daha sonra ***KOMŞU BEKLEME*** durumuna geçmektedir.

***KOMŞU BEKLEME*** durumunda ***m*** düğümü herhangi bir ***u*** komşusundan $\Gamma^{tree}_u$ kümesini alırsa $updateWr(\Gamma^{tree}_u)$ metodunu çalıştırmaktadır.

***KOMŞU BEKLEME*** durumunda ***m*** düğümü tüm Dtee komşularından $\Gamma^{tree}_u$ kümelerini alan ***m*** düğümü $(m, wr_m, partner_m)$ mesajını Dtor komşularına göndermektedir. Ağaçlar tarafından en küçük ağırlık oranına sahip düğüm olarak seçilip seçilmediği bilgisini tutan $isSelected_m$ değerini ve ağaçları birbirine bağlayıp bağlamayacağı bilgisini tutan $willMerge_m$ değerini True olarak başlatmaktadır. Ardından ***DTOR KARARLARI BEKLEME*** durumuna geçmektedir.

***WR BEKLEME*** durumunda ***m*** düğümü $(u, wr_u, partner_u)$ mesajı aldığında $wr_u$ ile $minWr_m$ değerini kıyaslamaktadır. $wr_u$ değeri küçük ise $minWr_m$ değerini $wr_u$ ile $minWrID_m$ değerini ***u*** ile, $minPartner_m$ değerini $partner_u$ ile güncellemektedir. Döngüsel bekleme oluşumunu engellemek için ağırlık oranlarındaki eşitlik durumunda ID'si veya partnerinin ID'si en büyük olan düğüm seçilmektedir.

***WR BEKLEME*** durumunda ***m*** düğümü tüm Dtee komşularından ve çocuklarından $(u, wr_u, partner_u)$ mesajlarını aldığında ***m*** düğümü kök düğüm değil

ise ***(minWrID$_m$, minWr$_m$, minPartner$_m$)*** mesajını ***parent*** düğümüne göndermektedir ve ***AĞAÇ KARARI BEKLEME*** durumuna geçmektedir. ***m*** düğümü kök düğüm ise ***minWr$_m$*** değerine bakmaktadır. ***minWr$_m$*** değeri sonsuz ise ağaca eklenecek yeni bir Dtee düğüme gerek kalmamıştır. ***FINISH*** mesajını Dtee komşularına ve çocuklarına göndermektedir ve ***DTOR*** durumuna geçerek algoritmayı sonlandırmaktadır. ***minWr$_m$*** değeri sonsuz değil ise ***minWrID$_m$*** değerini Dtee komşularına ve çocuklarına göndermektedir ve ***MERGE BEKLEME*** durumuna geçmektedir.

***AĞAÇ KARARI BEKLEME*** durumunda ***m*** düğümü ***parent*** düğümünden ***minWrID$_u$*** değerini alınca ***minWrID$_u$*** değerini ***minWrID$_u$*** değeri ile değiştirmektedir ve ***minWrID$_m$*** değerini DTEE komşularına ve çocuklarına göndermektedir. Daha sonra ***MERGE BEKLEME*** durumuna geçmektedir.

***FINISH*** mesajını aldığında ***FINISH*** mesajını DTEE komşularına ve ***$\Gamma^{children}$***'a göndermektedir. Daha sonra ***DTOR*** durumuna geçerek algoritmayı sonlandırmaktadır.

***DTOR KARARLARI BEKLEME*** durumunda ***m*** düğümü herhangi bir ***u*** komşusundan ***minWrID$_u$*** değerini aldığında ***minWrID$_u$*** değeri ***m*** değil ise komşu ağacı tarafından seçilmediğini, birleştirme işlemi yapamayacağını anlamaktadır ve ***isSelected$_m$*** değeri ile ***willMerge$_m$*** değerini False olarak değiştirmektedir. ***FINISH*** mesajını aldığında ***DTEE*** durumuna geçerek algoritmayı sonlandırmaktadır.

***DTOR KARARLARI BEKLEME*** durumunda ***m*** düğümü ***partner$_m$*** düğümünden ***(u, partner$_u$, isSelected$_u$)*** mesajı aldığında ***partner$_u$*** değeri ***m*** değil ise ***u*** düğümünün ***m*** düğümünü partner olarak seçmediğini anlar ve bağlama işlemi gerçekleştiremeyeceği için ***willMerge$_m$*** değerini False ile değiştirmektedir. ***isSelected$_u$*** değeri False ise ***m*** düğümü partnerinin komşu ağaçları tarafından seçilemediğini anlar ve bağlama işlemi gerçekleştiremeyeceği için ***willMerge$_m$*** değerini False ile değiştirmektedir.

***DTOR KARARLARI BEKLEME*** durumunda ***m*** düğümü tüm DTOR komşularından ***minWrID*** mesajlarını aldığında ***(m, partner$_m$, isSelected$_m$)*** mesajını DTEE komşularına göndermektedir ve ***PARTNER BEKLEME*** durumuna geçmektedir.

***PARTNER BEKLEME*** durumunda ***m*** düğümü herhangi bir ***u*** komşusundan ***I_AM_DTOR$_u$*** mesajı alırsa ***state$_u$*** değerini DTOR yapmaktadır.

***PARTNER BEKLEME*** durumunda ***m*** düğümü ***partner$_m$*** düğümünden ***(u, partner$_u$, isSelected$_u$)*** mesajı aldığında ***partner$_u$*** değeri ***m*** değil ise ***u*** düğümünün ***m*** düğümünü partner olarak seçmediğini anlar ve bağlama işlemi gerçekleştiremeyeceği için ***willMerge$_m$*** değerini False ile değiştirmektedir. ***isSelected$_u$*** değeri False ise ***m*** düğümü partnerinin komşu ağaçları tarafından seçilemediğini anlar ve bağlama işlemi gerçekleştiremeyeceği için ***willMerge$_m$*** değerini False ile değiştirmektedir.

***PARTNER BEKLEME*** durumunda ***m*** düğümü tüm Dtee komşularından ***(u, partner$_u$, isSelected$_u$)*** mesajlarını aldığında ***willMerge$_m$*** değeri False ise ***minWrID*** mesajlarında ***m*** düğümünü gönderen her komşu ağaçtan herhangi bir Dtor düğüme, bağlama işlemini gerçekleştiremeyeceği için ***NOT_MERGE$_m$*** mesajını göndermektedir. Ardından ***DTEE*** durumuna geçmektedir.

***PARTNER BEKLEME*** durumunda ***m*** düğümü tüm Dtee komşularından ***(u, partner$_u$, isSelected$_u$)*** mesajlarını aldığında ***willMerge$_m$*** değeri True ise kendisi ve varsa partneri ile birlikte komşu ağaçlarını birleştirmesi gerektiğini anlamaktadır. ***state$_m$*** değerini Dtor olarak değiştirmektedir. Dtor olduğunu komşularına bildirmek için ***I_AM_DTOR$_m$*** mesajını tüm komşularına göndermektedir. ***m partner$_m$***'den büyük ise ***treeID$_m$*** değerini ***m*** ile değiştirmektedir, ***partner$_m$***'i çocuklarına eklemektedir ve her komşu ağaçtan bir Dtor düğümü çocuklarına eklemektedir. ***partner$_m$*** değeri ***m***'den büyük ise ***treeID$_m$*** ve ***parent$_m$*** değerlerini ***partner$_m$*** ile değiştirmektedir ve ***partner$_m$***'nin komşusu olmayan her komşu ağaçtan bir Dtor düğümü çocuklarına eklemektedir. İçerisinde ***treeID$_m$*** değeri bulunan ***MERGE$_m$*** mesajını tüm çocuklarına göndermektedir. ***treeID$_m$*** değerini Dtee komşularına göndermektedir. ***minWr$_m$***, ***minWrID$_m$*** ve ***minPartner$_m$*** değerlerini sonsuza eşitlemektedir. Daha sonra ***WR BEKLEME*** durumuna geçmektedir.

***MERGE BEKLEME*** durumunda ***m*** düğümü herhangi bir ***u*** komşusundan ***I_AM_DTOR$_u$*** mesajı alırsa ***state$_u$*** değerini Dtor olarak değiştirmektedir.

***MERGE BEKLEME*** durumunda ***m*** düğümü ***NOT_MERGE$_u$*** mesajı aldığında ***m*** düğümü bağlanma işleminin yapılamayacağını anlamaktadır ve

çocuklarına ve $parent_m$ düğümüne bağlanma işleminin olmayacağını bildirmek için $NOT\_MERGE_m$ mesajlarını göndermektedir. $treeID_m$ değerini Dtee komşularına göndermektedir. $minWr_m$, $minWrID_m$ ve $minPartner_m$ değerlerini sonsuza eşitlemektedir. Ardından ***WR BEKLEME*** durumuna geçmektedir.

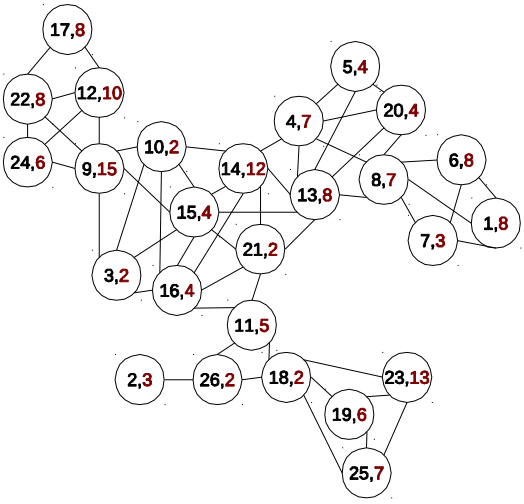
***MERGE BEKLEME*** durumunda ***m*** düğümü ***u*** düğümünden $MERGE_u$ mesajını aldığında ***m*** düğümü birleşme işleminin gerçekleştiğini anlamaktadır. $treeID_m$ değerini $MERGE_u \rightarrow treeID$ ile değiştirmektedir. $parent_m$'i çocuklarına eklemektedir. Çocuklarından ***u***'yu çıkarmaktadır. $parent_m$ değerini ***u*** ile değiştirmektedir. $MERGE_m$ mesajını tüm çocuklarına göndermektedir. $treeID_m$ değerini DTEE komşularına göndermektedir. $minWr_m$, $minWrID_m$ ve $minPartner_m$ değerlerini sonsuza eşitlemektedir. Daha sonra ***WR BEKLEME*** durumuna geçmektedir.

***updateWr($\Gamma^{tree}_u$)*** metodunda ***m*** düğümü ağırlık oranını güncellemektedir. $wr_{temp}$ değerini $(w_m+w_u)/\mathrm{C}\left(\left|\Gamma_m^{tree} \cup \Gamma_u^{tree}\right|, 2\right)$ şeklinde hesaplamaktadır. $wr_{temp}$ değeri $wr_m$ değerinden küçük ise $wr_m$ değerini $wr_{temp}$ değeri ile, $partner_m$ değerini ***u*** değeri ile değiştirmektedir. Eşitlik durumunda ise ID'si büyük olan tercih edilmektedir.

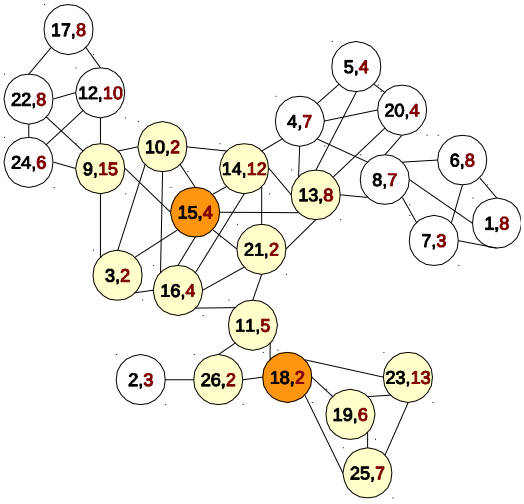
### 6.3.3 CN-MWCDS Algoritmasının Örnek Üzerinde Gösterimi

Şekil 6.5'te CN-MWCDS algoritmasının Şekil 6.1 ve Şekil 6.2'deki çizge üzerindeki adımları verilmiştir. Şekilde beyaz ile gösterilen düğümler IDLE, turuncu ile gösterilenler DTOR ve sarı ile gösterilenler DTEE düğümlerdir. Düğümler üzerinde siyah renkte ID'leri, kırmızı renkte ağırlıkları verilmiştir.

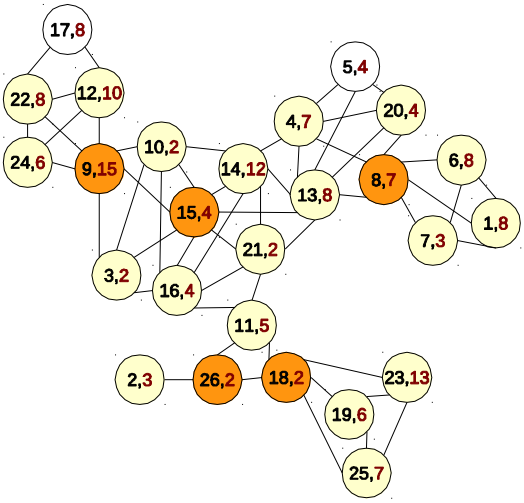
Algoritmanın başlangıcında tüm düğümler IDLE ve aktif durumdadır. Bu yüzden düğümlerin IDLE komşu sayılarını gösteren ***I*** değerleri komşu sayılarına eşittir. Örneğin 15 numaralı düğüm için $I_{15} = 7, 10$ numaralı düğüm için $I_{10} = 5$ ve $2$ numaralı düğüm için $I_2 = 1$ olmaktadır. Tüm düğümler ***I*** değerlerini tüm aktif komşularına göndermektedir. Tüm aktif komşularından ***I*** değerlerini alan düğümler her aktif komşusu için ağırlık oranlarını hesaplamaktadır. Örneğin ***15*** düğümü için yapılan hesaplamalar Tablo 6.1'de verilmiştir.



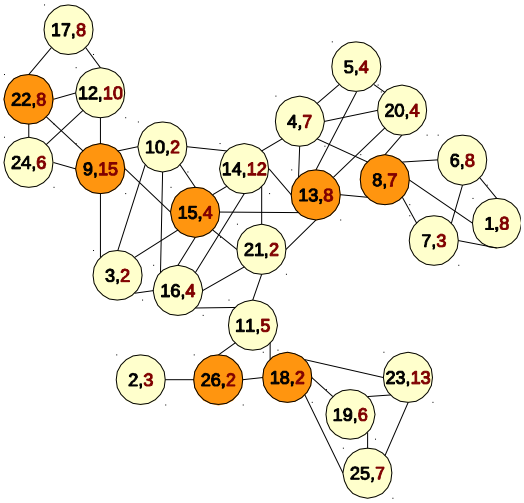

(a) Başlangıç durumu



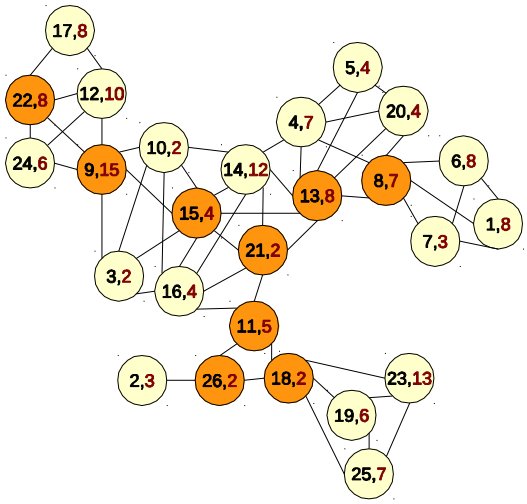

(b) Birinci tur sonunda çizgenin durumu




(c) İkinci tur sonunda çizgenin durumu




(d) Üçüncü tur sonunda çizgenin durumu




(e) STA sonunda çizgenin durumu

Şekil 6.5: CN-MWCDS algoritmasının örnek üzerinde gösterimi.

Tablo 6.1: Şekil 6.5'teki 15 numaralı düğümün ağırlık oranları.

| $u$ | $\Gamma_{15} \setminus \Gamma_u$ | $\Gamma_u \setminus \Gamma_{15}$ | $wr_{15,u}$ | $wr_{u,15}$ | $wr_{15,u} <= wr_{u,15}$ |
|---|---|---|---|---|---|
| ***3*** | {13,14,21} | { } | 4/(7*(3+1))=0.14 | 2/(4*(0+1))=0.5 | EVET |
| ***9*** | {13,14,16,21} | {12,22,24} | 4/(7*(4+1))=0.11 | 15/(6*(3+1))=0.62 | EVET |
| ***10*** | {13,21} | { } | 4/(7*(2+1))=0.19 | 2/(5*(0+1))=0.4 | EVET |
| ***13*** | {3,9,10,16} | {4,5,8,20} | 4/(7*(4+1))=0.11 | 8/(7*(4+1))=0.22 | EVET |
| ***14*** | {3,9} | {4} | 4/(7*(2+1))=0.19 | 12/(6*1+1)=1 | EVET |
| ***16*** | {9,14} | {11} | 4/(7*(2+1))=0.19 | 4/(6*1+1)=0.33 | EVET |
| ***21*** | {3,9,10} | {11} | 4/(7*(3+1))=0.14 | 2/(5*1+1)=0.2 | EVET |

Tablo 6.1'de gösterilen ağırlık oranlarına göre ***15*** numaralı düğüm yedi komşusundan da daha küçük ağırlık oranına sahiptir. ***15*** numaralı düğümün ağırlık oranları yedi komşusuna göre daha düşük olduğu için $Pwr_{15}$ değeri 7 olmaktadır. Tüm aktif düğümler ***Pwr*** değerlerini aktif komşularına göndermektedir.

Tüm aktif komşularından ***Pwr*** değerlerini alan DTEE düğümler kendisinin ve aktif komşularının ***Pwr*** değerleri içerisinden en büyüğüne sahip olan düğümün ID'sini ***MaxPwrID*** değerinde tutmaktadır ve tüm aktif komşularına göndermektedir. Örneğin ***3, 9, 10, 13, 14, 15, 16*** ve ***21*** numaralı düğümlerin ***MaxPwrID***'si ***15***'tir. Çünkü ***3, 9, 10, 13, 14, 15, 16*** ve ***21*** numaralı düğümlerin komşularının ***Pwr*** değerleri içerisinde ***15*** numaralı düğümün ***Pwr*** değeri 7 en büyük ***Pwr*** değeridir. ***15*** numaralı düğüm tüm komşularından ***MaxPwrID*** değerlerini aldığında hepsi ***15*** numaralı düğümü seçtiği için DTOR ve pasif olmaktadır. ***18*** numaralı düğüm de tüm komşularından ***MaxPwrID*** değerlerini aldığında ***MaxPwrID*** değerlerinin hepsi ***18*** olduğu için DTOR ve pasif olmaktadır. ***15*** ve ***18*** numaralı düğümler DTOR ve pasif olduklarını komşularına ***STATUS*** mesajları ile göndermektedir.

***15*** ve ***18*** numaralı düğümlerden ***STATUS*** mesajlarını alan ***3, 9, 10, 11, 13, 14, 16, 19, 21, 23, 25*** ve ***26*** numaralı düğümler IDLE durumda oldukları ve komşuları DTOR olduğu için durumlarını DTEE olarak değiştirmektedir, ardından ***STATUS*** mesajlarını tüm komşularına göndermektedir. ***MaxPwrID*** komşusundan ***STATUS*** mesajları alan tüm düğümler ***STATUS*** mesajlarını göndermektedir.

Tüm aktif komşularından ***STATUS*** mesajlarını alan düğümler ikinci tura başlamaktadır. Birinci turdaki işlemler ikinci turda da yapılmaktadır ve ***8, 9,*** ve ***26*** numaralı düğümler DTOR olarak, ***1, 2, 4, 6, 7, 12, 20, 22*** ve ***24*** numaralı düğümler DTEE olarak seçilmektedir. İkinci turun sonunda ***4, 5, 12, 13, 17, 20*** ve ***22*** numaralı düğümler haricindeki tüm düğümler ya DTOR oldukları için ya da DTOR olma ihtimalleri kalmadığı için pasif hale geçmektedir.

Üçüncü turda ***13*** ve ***22*** numaralı düğümler DTOR olarak, ***5*** ve ***17*** numaralı düğümler DTEE olarak seçilmektedir. Çizge içerisindeki tüm düğümler pasif hale geçmektedir ve birinci faz sonlanmaktadır.

İkinci faz başlamadan önce DTOR olan ***8, 9, 13, 15, 18, 22*** ve ***26*** numaralı düğümler başlangıç ağaçlarını oluşturmak için GHS algoritmasını çalıştırmaktadır. GHS algoritmasının sonucunda ***8, 9, 13, 15*** ve ***22*** numaralı düğümler ***13*** numaralı ağacı, ***18*** ve ***26*** numaralı düğümler ***26*** numaralı ağacı oluşturmaktadır ve algoritmanın ikinci fazı başlamaktadır.

Algoritmanın ikinci fazı başladığında ***8, 9, 13, 15*** ve ***22*** numaralı düğümler treeID değerleri olan ***13***'ü, ***18*** ve ***26*** numaralı düğümler ***26***'yı komşularına göndermektedir.

Tüm DTOR komşularından ***treeID*** değerlerini alan bazı ***DTEE*** düğümlerin komşu ağaç kümeleri $\Gamma^{tree}_{11}=\{26\}$, $\Gamma^{tree}_{16}=\{13\}$ ve $\Gamma^{tree}_{22}=\{13\}$ şeklindedir. DTEE düğümler komşu ağaç kümelerini DTEE komşularına göndermektedir.

Tüm DTEE komşularından $\Gamma^{tree}$ kümelerini alan DTEE düğümler ağırlıklarını hesaplamaktadır. Örneğin ***11*** numaralı düğüm ***16*** ve ***21*** numaralı düğümlerden $\Gamma^{tree}_{16}$ ve $\Gamma^{tree}_{21}$ kümelerini aldıktan sonra ağırlık oranını hesaplamaktadır. ***11*** numaralı düğümün sadece bir komşu ağacı olduğu için kendi başına ağaçları bağlayamamaktadır ve bir komşusunu partner seçerek ağırlık oranını hesaplamaktadır. ***11*** numaralı düğüm ***16*** numaralı düğüm ile birleşerek $wr_{11}=(w_{11}+w_{16})/\ \mathrm{C}\left(\left|\Gamma^{\mathrm{tree}}_{11}\ \mathrm{U}\ \Gamma^{\mathrm{tree}}_{16}\right|,2\right)$ formülünü kullanarak ağırlık oranını $w_{11}=(5+4)/\ \mathrm{C}\left(\left|\{13,26\}\right|,2\right)=9$ bulmaktadır. Aynı şekilde ***11*** numaralı düğüm ***21*** numaralı düğüm ile birleşerek ağırlık oranını $wr_{11}=(5+2)/\ \mathrm{C}\left(\left|\{13,26\}\right|,2\right)=7$

bulmaktadır. ***21*** numaralı düğüm ile oluşturduğu ağırlık oranı daha düşük olduğu için ağırlık oranı olarak ***7***'yi kullanmaktadır ve ***21*** numaralı düğümü partner olarak seçmektedir. ***21*** ve ***16*** numaralı düğümlerde partner olarak ***11*** numaralı düğümü seçmektedir ve ağırlık oranları sırasıyla ***7*** ve ***9***'dur. Çizgedeki diğer tüm DTEE düğümlerin ağırlık oranları hiç ağaç bağlayamadıkları için sonsuz olmaktadır. Ağırlık oranlarını hesaplayan DTEE düğümler ***(ID,wr,partner)*** mesajlarını DTOR düğümlere göndermektedir.

DTOR düğümler DTEE komşularından ***(ID,wr,partner)*** mesajlarını aldıklarında ağırlık oranı en düşük olan düğüm bilgilerini ağacın köküne iletmektedir. Ağaçların kök düğümleri tüm çocuklarından ve DTEE komşularından ***(ID,wr,partner)*** mesajlarını aldıktan sonra en küçük ağırlık oranına sahip olan düğümün ID'sini (***minWrID***) bulmaktadır ve ağaçtaki tüm DTOR düğümler ***minWrID*** değerini DTEE komşularına göndermektedir. Örneğin ***13*** numaralı düğüm içinde bulunduğu ağacın kökü olduğu için ***13*** numaralı ağaca komşu olan DTEE düğümler içerisinden en küçük ağırlık oranına sahip olan ***21*** numaralı düğümü bulmaktadır ve ***13*** numaralı ağaç içerisindeki tüm DTOR düğümler ***minWrID*** ile ***21*** numaralı düğümü seçtiklerini DTEE komşularına göndermektedir. ***26*** numaralı ağaçtaki DTOR düğümler ise ***minWrID*** ile ***11*** düğümünü seçtiklerini DTEE komşularına göndermektedir.

Tüm DTOR komşularından ***minWrID*** mesajını alan DTEE düğümler, DTEE komşularına partner düğümlerini ve seçilip seçilmediklerini ***(ID,partner,isSelected)*** mesajı ile göndermektedir. Tüm DTEE komşularından ***(ID,partner,isSelected)*** mesajlarını alan DTEE düğümler partnerleri yoksa ve kendi başlarına seçildiler ise, veya partnerleri varsa ve partnerleri ile birlikte seçildilerse komşu ağaçları bağlamak için ***MERGE*** mesajı göndermektedir. Örneğin ***11, 16*** ve ***21*** numaralı düğümler DTOR komşularından ***minWrID*** değerlerini aldıklarında komşuları tarafından seçilip seçilmediklerini öğrenmektedir. ***11*** numaralı düğüm komşusu olan ***26*** numaralı ağaç tarafından, ***21*** numaralı düğüm komşusu olan ***13*** numaralı ağaç tarafından seçilmiştir. Fakat düğümler partnerlerinin seçilip seçilmediğini ve partner olarak kendilerini seçip seçmediğini bilmedikleri için ***(ID,partner,isSelected)*** mesajlarını beklemektedir. ***11*** numaralı düğüm ***(11,21,True)*** mesajını, ***16*** numaralı düğüm ***(16,11,False)*** mesajını ve ***21*** numaralı düğüm ***(21,11,True)*** mesajını tüm DTEE

komşularına göndermektedir. ***11*** numaralı düğüm ***(21,11,True)*** mesajını aldığında ***21*** numaralı düğümün partner olarak kendisini seçtiğini, ***21*** numaralı düğümün komşu ağaçları tarafından seçildiğini anlamaktadır ve kendisi de ***21*** numaralı düğümü partner olarak seçtiği ve komşu ağaçları tarafından seçildiği için ***21*** numaralı düğüm ile birlikte bağlama işlemini gerçekleştirmektedir.

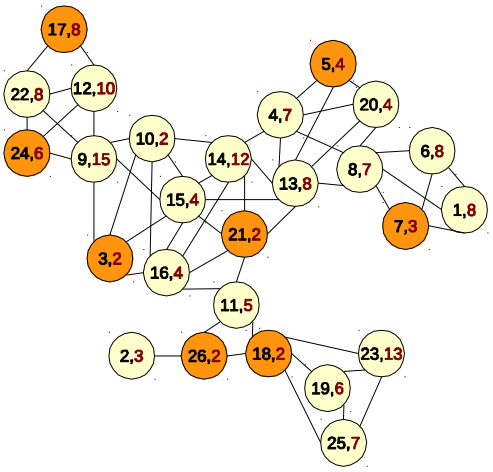
Bağlama işlemi gerçekleştirilirken ***11*** ve ***21*** numaralı düğümler DTOR olmaktadır ve ID'si büyük olan ***21*** numaralı düğüm kök düğüm olmaktadır. ***11*** numaralı düğüm ***21*** numaralı düğümü ebeveyni, ***21*** numaralı düğüm de ***11*** numaralı düğümü çocuğu yapmaktadır. ***11*** numaralı düğüm ***26*** numaralı ağaçtan bir düğümü, ***21*** numaralı düğüm ***13*** numaralı ağaçtan bir düğümü çocuklarına eklemektedir. Sonuç olarak ***13*** ve ***26*** numaralı ağaçtaki düğümler ***21*** numaralı ağaç üzerinde birleşerek tek bir ağaç oluşturmaktadır. Çizge üzerinde tek bir ağaç kaldığı için algoritma sonlanmaktadır.

WANG, ASYNSET-ASYNTREE ve CN-MWCDS algoritmalarının örnek gösterimleri aynı çizge üzerinde yapılmıştır. Şekil 6.6'da algoritmaların birinci ve ikinci fazlarının sonuçları verilmiştir. Birinci fazlarının sonuçlarına göre oluşan baskın kümelerin düğüm sayıları sırasıyla 8, 7 ve 7, oluşan baskın kümelerdeki düğümlerin ağırlıkları toplamları sırasıyla 27, 23 ve 46, birbirlerine komşu olan DTOR düğümlerin oluşturduğu ağaç sayıları 7, 6 ve 2'dir. Sonuçlardan anlaşıldığı üzere algoritmalar birbirlerine yakın sayıda düğüm seçmişlerdir fakat kümelerin toplam ağırlıklarına bakıldığı zaman CN-MWCDS algoritması diğer algoritmaların yaklaşık iki katı büyüklüğündedir. Bunun nedeni CN-MWCDS algoritmasının ilk fazının minimum bağlı baskın kümeden ziyade birbirlerine kolayca bağlanabilen bir baskın küme oluşturmayı hedeflemesidir. Sonuçlardan da görüldüğü üzere birinci fazlar sonucunda oluşan ağaç sayılarında CN-MWCDS algoritması diğer algoritmalara göre 3 kat daha az ağaç oluşturmuştur. Birinci fazların tamamlanması için gereken tur sayıları sırasıyla 2, 4 ve 3'tür. Tur sayıları kıyaslandığında WANG algoritmasının birinci fazı diğer algoritmalardan 2 kat daha hızlı sonuçlanmıştır.

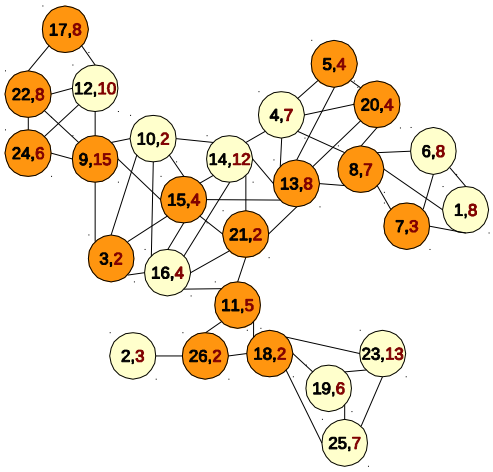
WANG, ASYNSET-ASYNTREE ve CN-MWCDS algoritmalarının ikinci fazda seçtikleri düğüm sayıları sırasıyla 7, 5 ve 2, seçilen düğümlerin toplam ağırlıkları sırasıyla 51, 39 ve 7, fazların sonlanması için gereken tur sayıları 6, 7 ve 1'dir. Sonuçlara bakıldığında CN-MWCDS algoritması diğer algoritmalara göre daha

hızlı çalışarak daha düşük maliyetli ve daha düşük ağırlık oranına sahip düğümler seçmiştir.

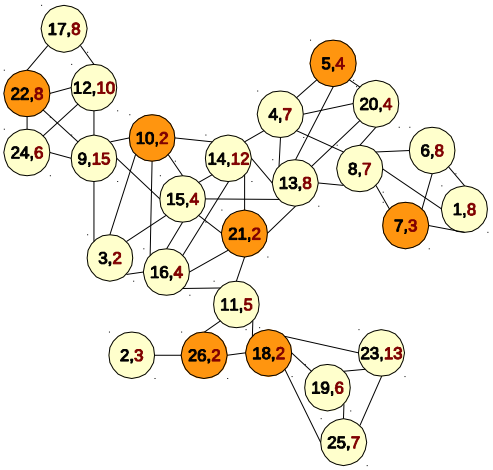
Elde edilen genel sonuçlara göre algoritmalar karşılaştırıldığında bağlı baskın kümeler içerisindeki düğüm sayısı sırasıyla 15, 12 ve 9, oluşan kümelerin ağırlıkları toplamı ise sırasıyla 78, 62 ve 53 olmaktadır. Sonuç olarak CDS-MWCDS algoritması diğer algoritmalara göre daha az düğüm ile daha az ağırlık oranına sahip bağlı baskın küme oluşturmuştur.



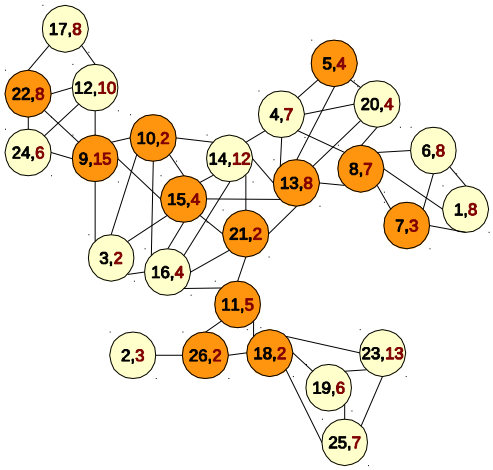

(a) WANG birinci faz sonunda çizgenin durumu



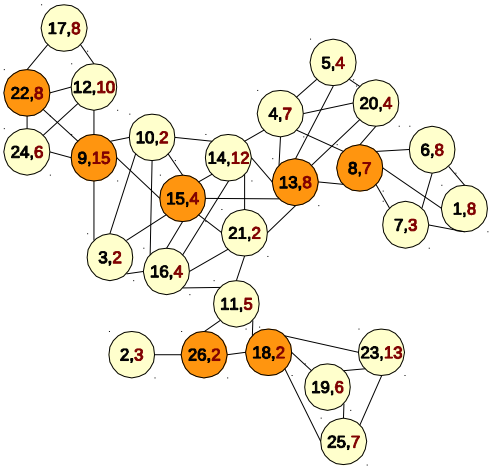

(b) WANG ikinci faz sonunda çizgenin durumu



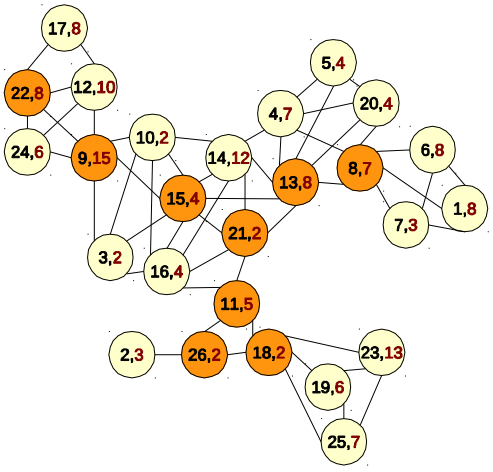

(c) ASYNSET sonunda çizgenin durumu




(d) ASYNTREE sonunda çizgenin durumu




(e) CNM-MWCDS birinci faz sonunda çizgenin durumu




(f) CNB-MWCDS ikinci faz sonunda çizgenin durumu

Şekil 6.6: Wang, ASYNTREE-ASYNSET ve CN-MWCDS algoritmalarının faz karşılaştırması.

## 6.4 WANG, ASYNSET-ASYNTREE ve CN-MWCDS Algoritmalarının Benzetim Sonuçlarının Karşılaştırılması

WANG, ASYNSET-ASYNTREE ve CN-MWCDS algoritmaları Contiki (Dunkels ve diğ. 2004) işletim sistemi üzerinde çalışacak şekilde kodlanmıştır. Benzetim ortamı olarak ise COOJA (Österlind ve diğ. 2006) kullanılmıştır. Algoritmaların benzetimi yapılırken ağ topolojilerini temsil eden, 30, 50 ve 80 düğümden oluşan çizgeler kullanılmıştır. Çizgeler tamamen rastgele oluşturulmuş birim disk çizgeleridir.

Tablo 6.2'de WANG algoritmasının benzetim sonuçları, Tablo 6.3'te ASYNSET-ASYNTREE algoritmasının benzetim sonuçları verilmiştir, Tablo 6.4'te ise CN-MWCDS algoritmasının benzetim sonuçları verilmiştir. Tablolardaki sonuçlar rastgele oluşturulan çizgeler üzerinde çalıştırılan algoritma sonuçlarının ortalamalarıdır. Tablodaki Toplam DTOR Sayısı, algoritma çalıştıktan sonra oluşan CDS içerisindeki düğüm sayısıdır. Toplam CDS Ağırlığı, oluşan CDS içerisindeki düğümlerin ağırlıklarının toplamıdır. Toplam Mesaj Sayısı, algoritmanın çalıştığı süre boyunca kullandığı mesaj sayısıdır. Toplam Geçen Süre, algoritmanın başladığı andan itibaren bittiği ana kadar geçen zamanın saniye olarak gösterimidir.

Tablo 6.2: WANG algoritmasının benzetim sonuçları.

| Çizgedeki Düğüm Sayısı | Toplam DTOR Sayısı | Toplam CDS Ağırlığı | Toplam Mesaj Sayısı | Toplam Geçen süre |
|---|---|---|---|---|
| 30 | 16,02083333 | 79,91666667 | 517,9791667 | 1361,041667 |
| 50 | 27,4 | 149,2666667 | 953,3777778 | 2266,777778 |
| 80 | 41,24242424 | 201,6969697 | 1722,151515 | 3601,272727 |

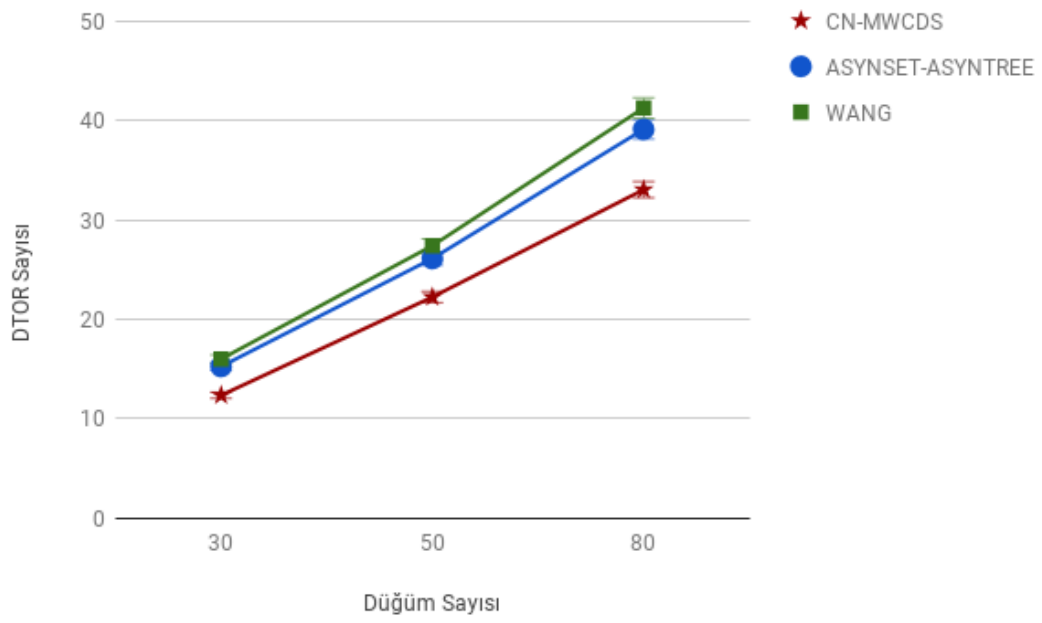
Tablo 6.3: ASYNSET-ASYNTREE algoritmasının benzetim sonuçları.

| Çizgedeki Düğüm Sayısı | Toplam DTOR Sayısı | Toplam CDS Ağırlığı | Toplam Mesaj Sayısı | Toplam Geçen süre |
|---|---|---|---|---|
| 30 | 15,27083333 | 76,5 | 3163,541667 | 4017,041667 |
| 50 | 26,11111111 | 143,6222222 | 7569,666667 | 10285,82222 |
| 80 | 39,09090909 | 191,2121212 | 17763,81818 | 19678,93939 |

Tablo 6.4: CN-MWCDS algoritmasının benzetim sonuçları.

| Çizgedeki Düğüm Sayısı | Toplam DTOR Sayısı | Toplam CDS Ağırlığı | Toplam Mesaj Sayısı | Toplam Geçen süre |
|---|---|---|---|---|
| 30 | 12,375 | 67,75 | 1989,375 | 2082,541667 |
| 50 | 22,24444444 | 130,6888889 | 4003,244444 | 3514,511111 |
| 80 | 33,03030303 | 177,0909091 | 8527,848485 | 9538,666667 |

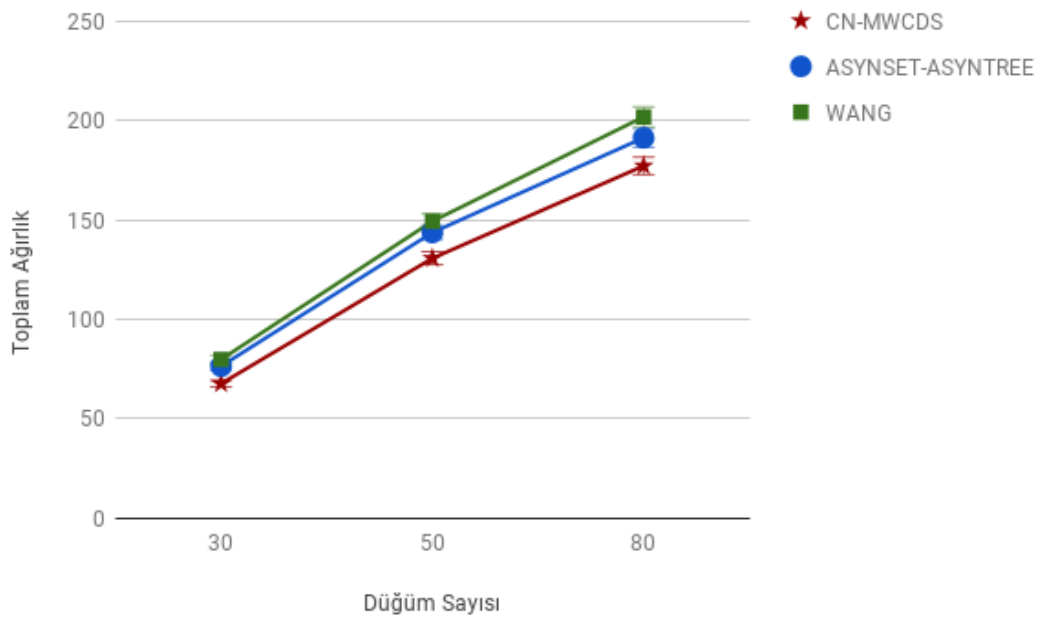
Şekil 6.7'de çizgede bulunan düğüm sayısına göre algoritmaların oluşturduğu CDS'lerin içerisindeki düğüm sayılarının grafiği verilmiştir. Sonuçlara göre CN-MWCDS algoritması WANG ve ASYNSET-ASYNTREE algoritmalarına göre daha düşük sayıda DTOR düğüm ile bağlı baskın küme oluşturmaktadır. CN-MWCDS algoritması ASYNSET-ASYNTREE algoritmasına göre 30 düğümlü çizgelerde %18, 50 düğümlü çizgelerde %14 ve 80 düğümlü çizgelerde %15 daha iyi performans göstermektedir. CN-MWCDS algoritması WANG algoritmasına göre ise 30 düğümlü çizgelerde %22, 50 düğümlü çizgelerde %18 ve 80 düğümlü çizgelerde %19 daha az düğüm kullanarak bağlı baskın küme oluşturmaktadır. Düğüm sayıları arasındaki fark ASYNSET-ASYNTREE algoritması ile %28'lere WANG algoritması ile ise %38'lere kadar çıkabilmektedir.




Şekil 6.7: CDS içerisindeki DTOR sayısı – düğüm sayısı grafiği.

Şekil 6.8'de çizgede bulunan düğüm sayısına göre algoritmaların oluşturduğu CDS'lerin içerisindeki düğümlerin toplam ağırlığının grafiği verilmiştir. Sonuçlara

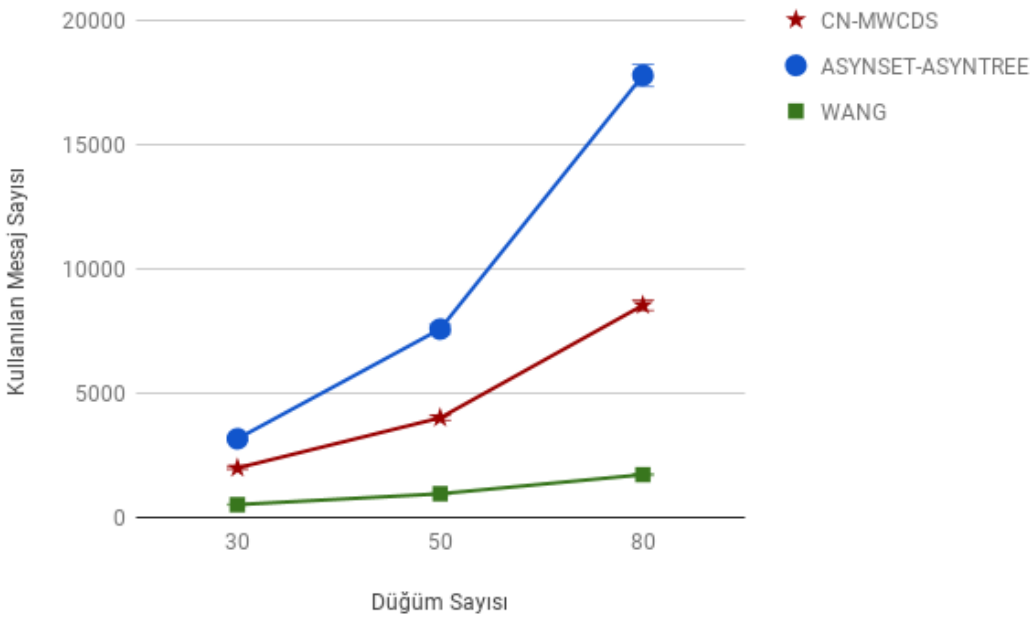
göre CN-MWCDS algoritması WANG ve ASYNSET-ASYNTREE algoritmalarına göre daha düşük toplam ağırlığa sahip bağlı baskın kümeler oluşturmaktadır. CN-MWCDS algoritması ASYNSET-ASYNTREE algoritmasına göre 30 düğümlü çizgelerde %11, 50 düğümlü çizgelerde %9 ve 80 düğümlü çizgelerde %8 daha iyi performans göstermektedir. CN-MWCDS algoritması WANG algoritmasına göre ise 30 düğümlü çizgelerde %15, 50 düğümlü çizgelerde %12 ve 80 düğümlü çizgelerde %12 daha az toplam ağırlık ile bağlı baskın küme oluşturmaktadır. Toplam ağırlıklar arasındaki fark ASYNSET-ASYNTREE algoritması ile %26'lara WANG algoritması ile ise %32'lere kadar çıkabilmektedir.




Şekil 6.8: CDS içerisindeki toplam ağırlık – düğüm sayısı grafiği.

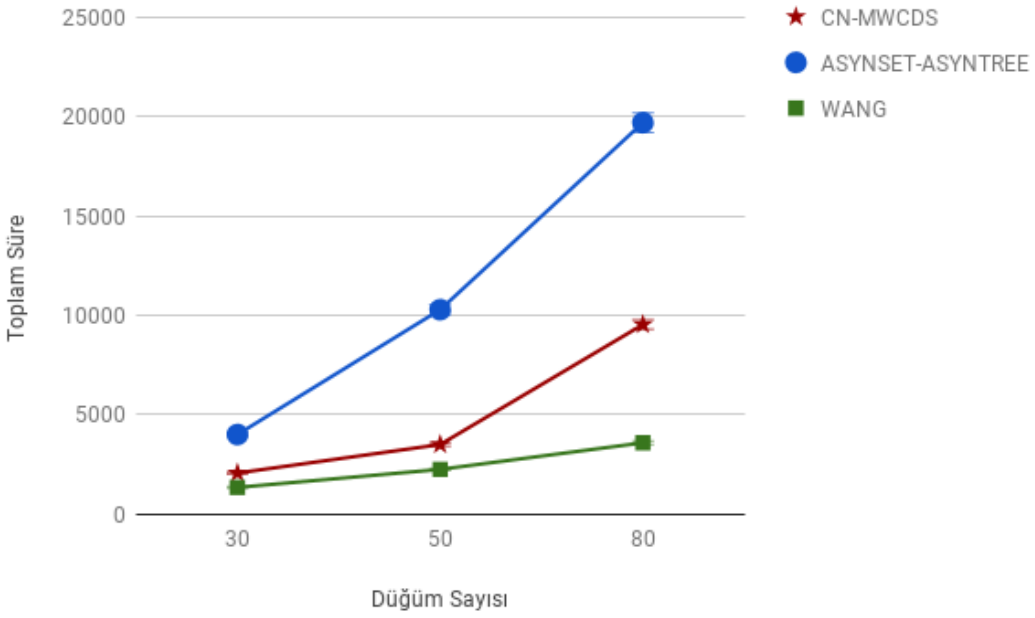
Şekil 6.9'da çizgede bulunan düğüm sayısına göre algoritmaların bağlı baskın küme oluştururken kullandıkları mesaj sayılarının grafiği verilmiştir. Sonuçlara göre CN-MWCDS algoritması ASYNSET-ASYNTREE algoritmasına göre, WANG algoritması ise hem CN-MWCDS hem de ASYNSET-ASYNTREE algoritmalarına göre daha düşük sayıda mesaj kullanmaktadır. CN-MWCDS algoritması ASYNSET-ASYNTREE algoritmasına göre 30 düğümlü çizgelerde %37, 50 düğümlü çizgelerde %47 ve 80 düğümlü çizgelerde %51 daha az mesaj kullanmaktadır. CN-MWCDS algoritması WANG algoritmasına göre ise 30 düğümlü çizgelerde %73, 50 düğümlü çizgelerde %76 ve 80 düğümlü çizgelerde %79 daha fazla mesaj kullanarak bağlı

baskın küme oluşturmaktadır. Toplam mesajlar arasındaki fark ASYNSET-ASYNTREE algoritması ile %63'lere WANG algoritması ile ise %83'lere kadar çıkabilmektedir.




Şekil 6.9: CDS oluşturulurken kullanılan toplam mesaj sayısı – düğüm sayısı grafiği.

Şekil 6.10'da çizgede bulunan düğüm sayısına göre algoritmaların bağlı baskın küme oluştururken harcadıkları sürelerin grafiği verilmiştir. Sonuçlara göre CN-MWCDS algoritması ASYNSET-ASYNTREE algoritmasına göre, WANG algoritması ise hem CN-MWCDS hem de ASYNSET-ASYNTREE algoritmalarına göre daha kısa süre harcamaktadır. CN-MWCDS algoritması ASYNSET-ASYNTREE algoritmasına göre 30 düğümlü çizgelerde %48, 50 düğümlü çizgelerde %65 ve 80 düğümlü çizgelerde %51 daha az süreye ihtiyaç duymaktadır. CN-MWCDS algoritması WANG algoritmasına göre ise 30 düğümlü çizgelerde %34, 50 düğümlü çizgelerde %35 ve 80 düğümlü çizgelerde %62 daha fazla süre harcayarak bağlı baskın küme oluşturmaktadır. Toplam harcanan süreler arasındaki fark ASYNSET-ASYNTREE algoritması ile %76'lara WANG algoritması ile ise %77'lere kadar çıkabilmektedir.




Şekil 6.10: CDS oluşturulurken harcanan süre – düğüm sayısı grafiği.

Sonuç olarak CN-MWCDS algoritması WANG ve ASYNSET-ASYNTREE algoritmalarına göre daha az sayıda düğüm kullanarak, daha küçük toplam ağırlıklı bağlı baskın kümeler oluşturmaktadır. Bağlı baskın kümeleri oluştururken CN-MWCDS algoritması ASYNSET-ASYNTREE algoritmasından, WANG algoritması ise CN-MWCDS algoritmasından daha az mesaj kullanarak daha az süre harcamaktadır.

# 7. SONUÇ VE ÖNERİLER

Telsiz duyarga ağları askeri, sağlık, gıda, tarım vb. endüstrilerinde sıkça kullanılmaktadır. Telsiz duyarga ağları, telsiz bir şekilde birbirlerine bağlanmış duyarga düğümlerinden oluşmaktadır. Düğümler enerji kaynağı olarak genellikle pil kullanmaktadır ve pillerden elde edilen enerji kısıtlı olduğu için enerjiyi verimli kullanmak önemli bir konu olmaktadır. Telsiz duyarga ağlarında düğümler enerjilerini en çok birbirleri arasında iletişim gerçekleştirdikçe harcamaktadır. Dolayısıyla düğümler arasındaki iletişim maliyeti düştükçe düğümlerin harcadıkları enerji miktarı da azalmaktadır. Telsiz duyarga ağlarında iletişim maliyetini düşürmek için bağlı baskın kümeler kullanılmaktadır. Bu çalışmada iki yeni bağlı baskın küme algoritması geliştirilmiştir. Bu algoritmalardan ilki Minimal Ağırlık ve Yönlendirme Maliyetli Bağlı Baskın Küme (MWOC-CDS) algoritmasıdır. İkincisi ise Kritik Düğüm Tabanlı Minimal Ağırlıklı Bağlı Baskın Küme (CN-MWCDS) algoritmasıdır.

MWOC-CDS algoritması düğümler arasında bağlı baskın küme üzerinde bulunan yol ağırlıklarını ve bağlı baskın kümeyi oluşturan düğümlerin ağırlıklarını azaltmayı hedeflemektedir. MWOC-CDS algoritması yönlendirme maliyetini minimuma düşüren MOC-CDS algoritmasının hesaba katmadığı düğümlerin ağırlıklarını da göz önünde bulundurarak CDS'i oluşturan düğümleri seçmektedir. MWOC-CDS ve MOC-CDS algoritmalarının benzetim sonuçları kıyaslandığında MWOC-CDS algoritmasının oluşturduğu CDS'lerin toplam ağırlıkları, düğüm sayıları ve küme içerisinden geçen en kısa ağırlıklı yolların toplam ağırlıkları önemli ölçüde düşerken, küme içerisinden geçen en kısa yolların azaldığı gözlemlenmiştir.

Bu çalışmanın devamında geliştirilen MWOC-CDS algoritmasının mesaj ve zaman karmaşıklıkları hesaplanabilir. Benzetim sonuçlarındaki performansı yaklaşım oranı hesaplanarak teorik kanıtlar ile desteklenebilir.

CN-MWCDS algoritması minimal ağırlıklı bir bağlı baskın küme oluşturmayı hedeflemektedir. CN-MWCDS algoritması iki fazlı bir algoritmadır. İlk fazında baskın kümeyi oluştururken, komşularına göre daha farklı düğümlere bağlanabilen

düğümleri seçmektedir. İkinci fazında ise Steiner Ağacı yaklaşım algoritması çalıştırarak baskın küme bağlı baskın kümeye dönüştürülmektedir. Geliştirilen algoritma WANG ve ASYNSET-ASYNTREE algoritmalarıyla karşılaştırılmıştır. Karşılaştırılan benzetim sonuçlarında oluşan bağlı baskın kümelerin toplam ağırlıkları ve düğüm sayıları diğer iki algoritmaya göre önemli ölçüde düştüğü, CDS'ler oluşturulurken kullanılan mesaj sayılarının ve harcanan sürelerin ASYNSET-ASYNTREE algoritmasına göre düştüğü, WANG algoritmasına göre ise arttığı gözlemlenmiştir.

Bu çalışmanın devamı olarak CN-MWCDS algoritmasının zaman ve mesaj karmaşıklıkları hesaplanabilir. Algoritmanın ikinci fazında partner olarak birden fazla düğüm seçilmesi düşünülebilir. Algoritmanın yaklaşım oranı (approximation ratio) hesaplanarak benzetim sonuçlarındaki performansı teorik sonuçlarla da desteklenebilir.

# 8. KAYNAKLAR


Abrach, H., Bhatti, S., Carlson, J., Dai, H., Rose, J., Sheth, A., Han, R., "MANTIS: System Support for multimodAl NeTworks of In-situ Sensors", In *Proceedings of the 2Nd ACM International Conference on Wireless Sensor Networks and Applications*, 50–59, New York, NY, USA: ACM. (2003).

ADVANTIC SISTEMAS, Y. S. S. L. (2018), "MTM-CM5000-SMA 802.15.4 Mote Modules | IoT Hardware Platforms", Retrieved June 4, 2018, from https://www.advanticsys.com/shop/mtmcm5000sma-p-23.html?zenid=e6b3f7dd29b4819ceea81e86a78e87fa?lang=,(2018).

Alzoubi, K. M., Wan, P.-J., & Frieder, O., "Message-optimal Connected Dominating Sets in Mobile Ad Hoc Networks", In *Proceedings of the 3rd ACM International Symposium on Mobile Ad Hoc Networking &Amp; Computing*, 157–164, New York, NY, USA: ACM., (2002).

Alzoubi, K. M., Wan, P.-J. W. P.-J., & Frieder, O., "New distributed algorithm for connected dominating set in wireless ad hoc networks", *Proceedings of the 35th Annual Hawaii International Conference on System Sciences*, *00*(c), 1–7., (2002).

Anastasi, G., Farruggia, O., Lo Re, G., & Ortolani, M., "Monitoring High-Quality Wine Production using Wireless Sensor Networks", *2009 42nd Hawaii International Conference on System Sciences*, 1–7, (2009).

Barr, R., Bicket, J. C., Dantas, D. S., Du, B., Kim, T. W. D., Zhou, B., & Sirer, E. G., "On the Need for System-level Support for Ad Hoc and Sensor Networks", *SIGOPS Oper. Syst. Rev.*, *36*(2), 1–5, (2002).

Butenko, S., Cheng, X., Oliveira, C. A., & Pardalos, P. M., "A New Heuristic for the Minimum Connected Dominating Set Problem on Ad Hoc Wireless Networks", In S. Butenko, R. Murphey, & P. M. Pardalos (Eds.), *Recent Developments in Cooperative Control and Optimization*, Boston, MA: Springer US.,61–73, (2004).

Cardei, M., Cheng, X., Cheng, X., & Du, D.-Z., "Connected Domination in Multihop Ad Hoc Wireless Networks", *Proceedings of the 6Th Joint Conference on Information Sciences*, 251–255, (2002).

Carnegie-Mellon Univ. Pittsburgh, P. A. D. of C. S., "*Proceedings of a Workshop on Distributed Sensor Nets: Held at Pittsburgh", PA on December 7-8*, (1978).

Cheng, X., & Ding, M., "An Approximation Algorithm for Connected Dominating Set in Ad Hoc Networks of the Health Sciences", *Proc. of International Workshop on Theoretical Aspects of Wireless Ad Hoc, Sensor, and Peerto- Peer Networks (TAWN)*, 1–5, (2004).

Chinchuluun, R., Lee, W. S., Bhorania, J., & Pardalos, P. M., "*Connected Dominating Set: Theory and Applications", Advances in Modeling Agricultural Systems* 25., (2009).

Chong, C. Y., & Kumar, S. P., "Sensor networks: Evolution, opportunities, and challenges", *Proceedings of the IEEE*, *91*(8), 1247–1256. (2003).

Clark, B. N., Colbourn, C. J., & Johnson, D. S., "Unit Disk Graphs. *Annals of Discrete Mathematics",* 165–177, (1991).

Dagdeviren, O., Erciyes, K., & Tse, S., "Semi-asynchronous and distributed weighted connected dominating set algorithms for wireless sensor networks", *Computer Standards and Interfaces*, *42*, 143–156, (2015).

Dai, F., & Wu, J., "An Extended Localized Algorithms for Connected Dominating Set Formation in Ad Hoc Wireless Networks", *IEEE Transactions on Parallel and Distributed Systems*, *15*(10), 908–920, (2004).

Dai, F., & Wu, J., "On constructing k-connected k-dominating set in wireless ad hoc and sensor networks", *Journal of Parallel and Distributed Computing*, *66*(7), 947–958, (2006).

Dai, F., Wu, J., & Raton, B., "On Constructing k -Connected k -Dominating Set in Wireless", *Parallel and Distributed Processing Symposium, Proceedings. 19th IEEE International*, *00*(c), (2005).

Das, B., & Bharghavan, V., "Routing in ad-hoc networks using minimum connected dominating sets", *Proceedings of ICC'97 - International Conference on Communications*, *1*, 376–380, (1997).

Ding, L., Gao, X., Wu, W., Lee, W., Zhu, X., & Du, D. Z., "Distributed construction of connected dominating sets with minimum routing cost in wireless networks", *Proceedings - International Conference on Distributed Computing Systems*, 448–457, (2010).

Du, D.-Z., & Pardalos, P. M. "*Handbook of combinatorial optimisation: supplement volume B"*, (2005).

Dunkels, A., Gronvall, B., & Voigt, T., "Contiki - a lightweight and flexible operating system for tiny networked sensors", In *29th Annual IEEE International Conference on Local Computer Networks*, 455–462, (2004).

Farooq, M. O., & Kunz, T., "Operating Systems for Wireless Sensor Networks: A Survey", *Sensors*, *11*(6), 5900–5930, (2011).

Gallager, R. G., Humblet, P. A., & Spira, P. M., "A Distributed Algorithm for Minimum-Weight Spanning Trees", *ACM Trans. Program. Lang. Syst.*, *5*(1), 66–77, (1983).

Garey, M. R., & Johnson, D. S., "Computers and Intractability: A Guide to the Theory of NP-Completeness (Series of Books in the Mathematical Sciences)", *Computers and Intractability*, 340, (1979).

Ghaffari, M., "Near-Optimal Distributed Approximation of Minimum-Weight Connected Dominating Set", In J. Esparza, P. Fraigniaud, T. Husfeldt, & E. Koutsoupias (Eds.), *Automata, Languages, and Programming*, Berlin, Heidelberg: Springer Berlin Heidelberg, 483–494, (2014).

Guha, S., & Khuller, S., "Approximation Algorithms for Connected Dominating Sets", *Algorithmica*, *20*(4), 374–387, (1998).

Healy, M., Newe, T., & Lewis, E., "Wireless sensor node hardware: A review", *Proceedings of IEEE Sensors*, 621–624, (2008).

Hong, S., & Kim, T. H. "Designing a State-driven Operating System for Dynamically Reconfigurable Sensor Networks", *Proceedings of the 2003 System on Chip SoC Design Conference*, (10006826), 40–42, (2003).

Jevtić, M., Zogović, N., & Dimić, G., "Evaluation of Wireless Sensor Network Simulators", *Proceedings of the 17th Telecommunications Forum TELFOR 2009 Belgrade Serbia*, 1303–1306, (2009).

Jian, T., "An O(20.304n) Algorithm for Solving Maximum Independent Set Problem", *IEEE Transactions on Computers*, *C-35*(9), 847–851, (1986).

K., H. F., & S., R. D., "Steiner tree problems", *Networks*, *22*(1), 55–89, (1992).

Kumar, S., & Shepherd, D., "SensIT: Sensor information technology for the warfighter", *Proc. 4th Int. Conf. on Information Fusion*, 1–7, (2001).

Levis, P., Lee, N., Welsh, M., & Culler, D., "TOSSIM: Accurate and Scalable Simulation of Entire TinyOS Applications", In *Proceedings of the 1st International Conference on Embedded Networked Sensor Systems*, New York, NY, USA: ACM, 126–137, (2003).

Levis, P., Madden, S., Polastre, J., Szewczyk, R., Whitehouse, K., Woo, A., Culler, D., "TinyOS: An Operating System for Sensor Networks", In W. Weber, J. M. Rabaey, & E. Aarts (Eds.), *Ambient Intelligence*, Berlin, Heidelberg: Springer Berlin Heidelberg, 115–148, (2005).

Li, X.-Y., "*Wireless Ad Hoc and Sensor Networks", Chemistry & …*. Retrieved from http://onlinelibrary.wiley.com/doi/10.1002/cbdv.200490137/abstract, (2008).

Mekni, M., & Moulin, B., "A survey on sensor webs simulation tools", *Proceedings - 2nd Int. Conf. Sensor Technol. Appl., SENSORCOMM 2008, Includes: MESH 2008 Conf. Mesh Networks; ENOPT 2008 Energy Optim. Wireless Sensors Networks, UNWAT 2008 Under Water Sensors Systems*, 574–579, (2008).

Misra, R., & Mandal, C., "Rotation of CDS via connected domatic partition in ad hoc sensor networks" *IEEE Transactions on Mobile Computing*, *8*(4), 488–499, (2009).

Misra, R., & Mandal, C., "a Collaborative Cover Heuristic for Ad Hoc Sensor

Networks", *IEEE Transaction on Parallel and Distributed Systems (TPDS)*, *21*(3), 292–302, (2010).

Mohanty, J. P., Mandal, C., & Reade, C., "Distributed construction of minimum Connected Dominating Set in wireless sensor network using two-hop information", *Computer Networks*, *123*, 137–152, (2017).

Mohapatra, S., & Kanungo, P., "Performance analysis of AODV, DSR, OLSR and DSDV routing protocols using NS2 simulator", *Procedia Engineering*, *30*(2011), 69–76, (2012).

Myers, C., & Oppenheim, A., "Knowledge Based Speech Analysis and Enhancement", *Acoustics, Speech, and …*, 1–4, (1984).

O'Donovan, T., O'Donoghue, J., Sreenan, C., Sammon, D., O'Reilly, P., & O'Connor, K. A. "A context aware wireless body area network (BAN)", *2009 3rd International Conference on Pervasive Computing Technologies for Healthcare*, 1–8, (2009).

Österlind, F., Dunkels, A., Eriksson, J., Finne, N., & Voigt, T., "Cross-level sensor network simulation with COOJA", *Proceedings - Conference on Local Computer Networks, LCN*, 641–648, (2006).

Shi, T., Cheng, S., Cai, Z., Li, Y., & Li, J., "Exploring Connected Dominating Sets in Energy Harvest Networks", *IEEE/ACM TRANSACTIONS ON NETWORKING*, *25*(3), 1803–1817, (2017).

Sohraby, K., Minoli, D., & Znati, T., "*Wireless Sensor Networks", Booksgooglecom*, (2007).

Tiwari, A., Ballal, P., & Lewis, F. L., "Energy-efficient Wireless Sensor Network Design and Implementation for Condition-based Maintenance", *ACM Trans. Sen. Netw.*, *3*(1), (2007).

Varga, A., & Hornig, R., "An Overview of the OMNeT++ Simulation Environment", In *Proceedings of the 1st International Conference on Simulation Tools and Techniques for Communications, Networks and Systems & Workshops*, ICST, Brussels, Belgium, Belgium: ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), 110, (2008).

Wang, Y., Wang, W., & Li, X.-Y., "Efficient Distributed Low-Cost Weighted Backbone Formation for Wireless Ad Hoc Networks", *IEEE Transaction on Parallel and Distributed Systems (TPDS)*, *17*(7), 681–693, (2006).

Wu, J., & Li, H., "On Calculating Connected Dominating Set for Efficient Routing in Ad Hoc Wireless Networks", In *Proceedings of the 3rd International Workshop on Discrete Algorithms and Methods for Mobile Computing and Communications*, New York, NY, USA: ACM, 7–14, (1999).

# 9. ÖZGEÇMİŞ

| | | |
|---|---|---|
| Adı Soyadı | : | Mustafa TOSUN |
| Doğum Yeri ve Tarihi | : | Akşehir / KONYA , 23/11/1992 |
| Lisans Üniversite | : | Pamukkale Üniversitesi – Bilgisayar Mühendisliği |
| Elektronik posta | : | ce.mustafatosun@gmail.com |

**Proje Listesi :**

• Haytaoğlu, E., Tosun M., "Dağıtık Sistemlerde Yeni Bir Minimal Ağırlıklı Bağlı Baskın Küme Algoritması", Pamukkale Üniversitesi Bilimsel Araştırma Projeleri Koordinasyon Birimi, Proje No:2018FEBE013

**Konferans listesi :**

• TOSUN, M., HAYTAOĞLU, E., GÜLEÇ, Ö., "A Minimal Weight and Routing Cost Connected Dominating Set Algorithm for Wireless Sensor Networks", 26. IEEE Signal Processing and Comminications Applications Conference, (Çeşme) (pp. 1-4), https://doi.org/10.1109/SIU.2018.8404579, (2018)

• TOSUN, M., HAYTAOĞLU, E., "A New Distributed Weighted Connected Dominating Set Algorithm for Wireless Sensor Networks", 8. IEEE International Conference on Consumer Electronics, (Berlin) (kabul edildi), (2018)