DOKUZ EYLÜL ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

SEVİYELERE DAYALI AĞIRLIKLI ORTALAMA TEMELLİ BULANIK ÇIKARSAMA YÖNTEMİYLE KABLOSUZ SENSÖR AĞLARINDA KÜME BAŞI BELİRLEME VE UYGULAMASI

Zülküf Tekin ERTEN

Nisan, 2019
İZMİR

SEVİYELERE DAYALI AĞIRLIKLI ORTALAMA TEMELLİ BULANIK ÇIKARSAMA YÖNTEMİYLE KABLOSUZ SENSÖR AĞLARINDA KÜME BAŞI BELİRLEME VE UYGULAMASI

Dokuz Eylül Üniversitesi Fen Bilimleri Enstitüsü
Yüksek Lisans Tezi
Bilgisayar Bilimleri Anabilim Dalı

Zülküf Tekin ERTEN

Nisan, 2019
İZMİR

YÜKSEK LİSANS TEZİ SINAV SONUÇ FORMU

ZÜLKÜF TEKİN ERTEN, tarafından **DR. ÖĞR. ÜYESİ RESMİYE NASİBOĞLU** yönetiminde hazırlanan "**SEVİYELERE DAYALI AĞIRLIKLI ORTALAMA TEMELLİ BULANIK ÇIKARSAMA YÖNTEMİYLE KABLOSUZ SENSÖR AĞLARINDA KÜME BAŞI BELİRLEME VE UYGULAMASI**" başlıklı tez tarafımızdan okunmuş, kapsamı ve niteliği açısından bir Yüksek Lisans tezi olarak kabul edilmiştir.

Dr.Öğr.Üyesi Resmiye NASİBOĞLU

Yönetici

Doç.Dr. Emel Kurnoğlu Karadeniz

Jüri Üyesi

Prof.Dr. Urfat NURİYEV

Jüri Üyesi

Prof.Dr. Kadriye ERTEKİN

Müdür

Fen Bilimleri Enstitüsü

TEŞEKKÜR

Tez çalışmalarım sırasında kıymetli bilgi, birikim ve tecrübeleri ile bana yol gösterici ve destek olan değerli danışman hocam sayın Dr.Öğr.Üyesi Resmiye NASİBOĞLU'na, ilgisini ve önerilerini göstermekten kaçınmayan Bilgisayar Bilimleri Ana Bilim Dalı Başkanı Sayın Prof. Dr. Efendi NASİBOĞLU'na sonsuz teşekkür ve saygılarımı sunarım.

Zülküf Tekin ERTEN

SEVİYELERE DAYALI AĞIRLIKLI ORTALAMA TEMELLİ BULANIK ÇIKARSAMA YÖNTEMİYLE KABLOSUZ SENSÖR AĞLARINDA KÜME BAŞI BELİRLEME VE UYGULAMASI

ÖZ

Bulanık mantık, günümüzün bilgisayar teknolojileri ve yapay zeka çalışmaları için vazgeçilmez bir unsurdur. Makinelerin sadece 0 ve 1'den ibaret olan dünyasını bulanık sayılar ve dilsel değişkenler vasıtası ile genişletmeyi başarmıştır. Bunun sonucunda birçok alanda bulanık mantık kullanılarak çözüm arayışlarına gidilmiştir.

Kablosuz sensör ağları bulanık mantığın kullanılabileceği alanlardan birisidir. Sensörlerin kümelendirilmesi için çeşitli yöntemler geliştirilmiştir. Bu yöntemlerin ortak amacı ağın ömrünü uzatarak maksimum fayda elde edebilmektir.

Bu çalışmada, sensörlerin kümelendirilmesinde Seviyelere Dayalı Ağırlıklı Ortalama (WABL) temelli Bulanık Çıkarım Sisteminin nasıl kullanılacağı açıklanmaktadır. Bulanık Çıkarım Sistemi sonucunda oluşan kompozit bulanık sayılarda WABL değerini analitik olarak hesaplayabilmek için yeni teorem ispat edilmiştir.

Sunulan örnek uygulamada, dilsel değişkenler kullanarak sensörlerin küme başı olma olasılıkları belirlenmekte ve bu olasılıklar çerçevesinde kümeler oluşturmaktadır. Kümelerin her bir döngüde nasıl değiştiği görülmektedir. Bu değişime uygun olarak gerekli durumlarda yeni küme başlıkları belirlenmektedir. Bunun sayesinde ağın ömrünün artırılmasına ve sensörlerden maksimum yararlanılmasına olanak sağlanmıştır.

Anahtar kelimeler: Kablosuz sensör ağı, bulanık mantık, WABL, kümelendirme

DETERMINATION OF CLUSTER HEAD VIA FUZZY INFERENCE SYSTEM USING WEIGHTED AVERAGING BASED ON LEVELS AND RELATED APPLICATION

ABSTRACT

Fuzzy logic is a dispensable element for computer technologies and artificial intelligence studies of present-day. Fuzzy logic has overcome to expand the world of machines by linguistic variables which only understand 0 and 1. As a result, fuzzy logic has been used in many areas in order to find solutions.

Wireless sensor networks are one of the fields in which fuzzy logic can be used. Various methods have been developed for clustering sensors. Common purpose of these methods is gaining maximum profit by prolonging the lifetime of networks.

In this study, how Fuzzy Inference System using Weighted Averaging Based on Levels (WABL) is used in clustering of sensors is explained. A new theorem has been proved in order to calculate the WABL value in composite fuzzy numbers resulting from the Fuzzy Inference System

In the example presented, linguistic variables are used to determine the probabilities of sensors to be the cluster head and the clusters within the framework of these possibilities are constructed. It is seen how the clusters change in each cycle. In accordance with this change, new cluster heads are determined when necessary. This has allowed the network to increase its life and maximum utilization of the sensors.

Keywords: Wireless sensor network, fuzzy logic, WABL, clustering

İÇİNDEKİLER

Sayfa

ŞEKİLLER LİSTESİ

Sayfa

TABLOLAR LİSTESİ

Sayfa

BÖLÜM BİR
GİRİŞ

1.1 Giriş

Bu tez çalışmasında, kablosuz bir sensör ağının ömrünün Seviyelere Dayalı Ağırlıklı Ortalama Temelli Bulanık Çıkarsama Yöntemi ile nasıl belirlendiğinin uygulamalı olarak gösterimi amaçlanmaktadır. Java programlama dili ile dinamik bir uygulama hazırlanmıştır. Düğüm noktası sayısı, küme başı düğüm noktalarının sayısı, düğüm noktalarının rastgele olarak yerleştirildiği alanın büyüklüğü, başlangıç enerji değerleri, gönderilen bit sayısı ve diğer parametreler değiştirilebilmektedir. Uygulama neticesinde alanın büyüklüğüne, düğüm noktası sayısına, küme başı düğüm noktası sayısına, kullanılan enerji miktarı ve ilgili parametrelere göre karşılaştırma olanağı sağlanmaktadır.

1.2 Tezin Organizasyonu

Bu tez 4 bölümden oluşmaktadır. Birinci bölümde bulanık mantık kavramı ve bulanık küme teorisi anlatılmaktadır. Bulanık Çıkarım Yöntemi tanımı ve Seviyelere Dayalı Ağırlıklı Ortalama Temelli Bulanık Çıkarsama Yöntemi ile ilgili bilgi verilmektedir.

İkinci bölümde; K-Ortalama Algoritması ile ilgili genel bilgiler verilmektedir. K-Ortalama Algoritmasının tanımı yapılmakta, verilen bir örnekle algoritmanın çalışma prensibi ile ilgili bilgi verilmektedir.

Üçüncü bölümde; kablosuz sensör ağları anlatılmaktadır. Kablosuz sensör ağlarının tanımı, kullanımı, yapısı ve enerji kullanımı ile ilgili bilgi verilmektedir. Kablosuz sensör ağı yapılarından hiyerarşik yapı anlatılmaktadır.

Dördüncü bölümde uygulama kısmı anlatılmaktadır. Bulanık mantık ile kablosuz sensör ağının küme başı düğüm noktalarını seçimi algoritması açıklamakta, yazılan programın aşamaları anlatılmaktadır.

1.3 Yapılan Çalışmalar

Bulanık mantık çıkarım sistemleri günümüzde her alanda olduğu gibi kablosuz sensör ağlarının etkinliğini arttırmak amacıyla da kullanılmaya başlanmıştır. Gupta, Riordan ve Sampalli (2005) yaptıkları çalışmada; kablosuz sensör ağlarının ömrünü hesaplamada kullanılan LEACH (Low Energy Adaptive Clustering Hierarchy) metodu ile bulanık mantık içeren bir yöntemi karşılaştırılmıştır. Çalışmalarında kriter olarak düğüm noktalarının enerji seviyeleri, merkeze yakınlıkları ve yoğunluk oranları kullanılmıştır. Sharma ve Kumar (2012) yaptıkları çalışmada; LEACH ve CHEF (Cluster Head Election Mechanism Using Fuzzy Logic) protokollerini karşılaştırmıştır. Kriter olarak pil ömrü ve küme başına uzaklık değerleri kullanarak MATLAB programı üzerinden simülasyon yapılmıştır. Alami ve Najid (2015) yaptıkları çalışmada; bulanık mantık kullanılarak kablosuz sensör ağlarının ömrünü LEACH ve SEP (Stable Election Protocol) protokolleri ile karşılaştırmıştır. Çalışmalarında kriter olarak düğüm noktalarının enerji seviyeleri, baz istasyonuna olan uzaklıkları ve diğer düğüm noktalarına olan uzaklıklarının toplamı kullanılmıştır. Sert, Yazıcı ve Dökeroğlu (2016) yaptıkları çalışmada; LEACH, CHEF ve DFCA (Distributed Fuzzy Clustering Algorithm) protokollerini karşılaştırmıştır. Çalışmalarında kriter olarak düğüm noktalarının enerji seviyeleri ve yoğunluk oranları kullanılmıştır. Mehra, Doja ve Alam (2018) bulanık mantığa dayalı bir modeli diğer modellerle karşılaştırmış ve bulanık mantığın daha etkin olduğu sonucuna varmıştır. Wang, Yang, Zheng, Zhang, ve Kim (2012) daha uzun ömürlü bir kablosuz sensor ağı için MHRP (Energy-Efficient Multi-hop Hierarchical Routing Protocol) yöntemini sunmuşlardır. Iancu (2012) Mamdani tipi bulanık mantık davranışlarını araştırmıştır Jain, Gupta ve Sinha (2013) çeşitli kümeleme protokolü önererek karşılaştırma yapmıştır. Muhammed (2016) çalışmasında, kablosuz algılayıcı ağlarda kümeleme mimarisini kullanarak daha yüksek enerji verimliliği sağlamayı amaçlamıştır. Najmeh (2015) çalışmasında, kablosuz sensör ağlarının toplam enerji sarfiyatını minimize edecek yeni bir yöntem üzerinde çalışmıştır.

Bulanık Çıkarım Sistemi (FIS), sensör ağı çalışmalarında yaygın olarak

kullanılmaktadır. Durulaştırma, bu sistemin önemli bir parçasıdır. Bulanık çıkarım sistemlerinde MOM (Mean of Maxima), COA (Centroid of Area), BOA (Bisector of Area) gibi durulaştırma operatörleri sıklıkla kullanılmaktadır. Biz çalışmamızda Weighted Averaging Based on Levels (WABL) yöntemini kullanacağız. WABL daha genel ve esnek bir durulaştırma yöntemidir. Mert (2006) çalışmasında, WABL ve MIN, MAX, MOM and COA yöntemleri arasındaki ilişkiyi teorik olarak araştırmıştır. Nasiboglu ve Abdullayeva (2018), ayrık yamuk bulanık sayılar için WABL konsepini araştırmış ve hesaplama kolaylığı sağlayacak bir analitik formülün ispatını yapmıştır. Nasibov and Mert (2007) çalışmalarında WABL, COA ve MOM yöntemlerinin karşılaştırmalı analizini yapmıştır.

BÖLÜM İKİ
BULANIK MANTIK

2.1 Bulanık Mantığın Tanımı

İnsan beyni makinelerden farklı çalışır. Makineler 0 ve 1 kodlarını kullanırlar ve sonuç iki değerden oluşur; doğru veya yanlış. İnsanlar için sadece doğru veya yanlış yoktur, ara değerler de vardır. Bir problemin çözümü tamamen yanlış değil de bir kısmı yanlış olabilir veya tamamı doğru değil de bir kısmı doğru olabilir. Makine bunu sadece 0 veya 1 ile ifade edemez.

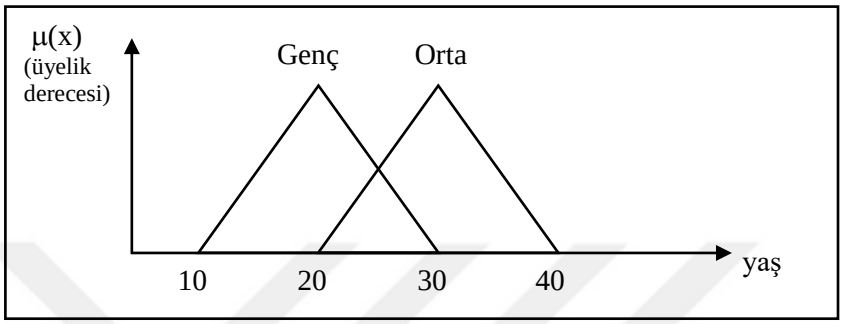
Bir cisim için uzun veya kısa gibi iki kesin ifade de kullanılmayabilir; çok uzun, çok kısa, biraz uzun, kısmen uzun olarak da tanımlanabilir. Klasik mantık ile bu ifadeleri makinelerin modellemesi imkânsızdır. Bu yüzden bulanık mantık kullanılmaktadır.

Günlük konuşma dilini kullanan bulanık mantık, dilsel değişkenler yardımıyla insan mantığına en yakın çözümleri bulabilir. Bir insanın yaşı sorulduğunda insan mantığına göre onun çok genç, genç, orta yaşlı, yaşlı veya çok yaşlı olduğu söylenebilir. Bulanık mantık ile yaş aralıkları üyelik dereceleri ile bu dilsel değişkenlere tanımlandığında makine bize o insanın hangi gruba ait olduğunu rahatlıkla söyleyebilmektedir.

2.2 Bulanık Küme Teorisi

Klasik mantığa göre bir değer bir kümenin elemanıdır veya değildir. Bulanık mantıkta ise bir değerin üyelik derecesine göre elemanı olup olmadığı ifade edilir. Örnek olarak; 30 yaşındaki bir insan klasik mantığa göre orta yaşlıdır; 29 yaşındaki bir insan ise gençtir, orta yaşlı değildir. 20 yaşındaki biri de gençtir ve klasik mantığa göre ikisi de aynı kümeye aittir ve bir fark belirtilememektedir.

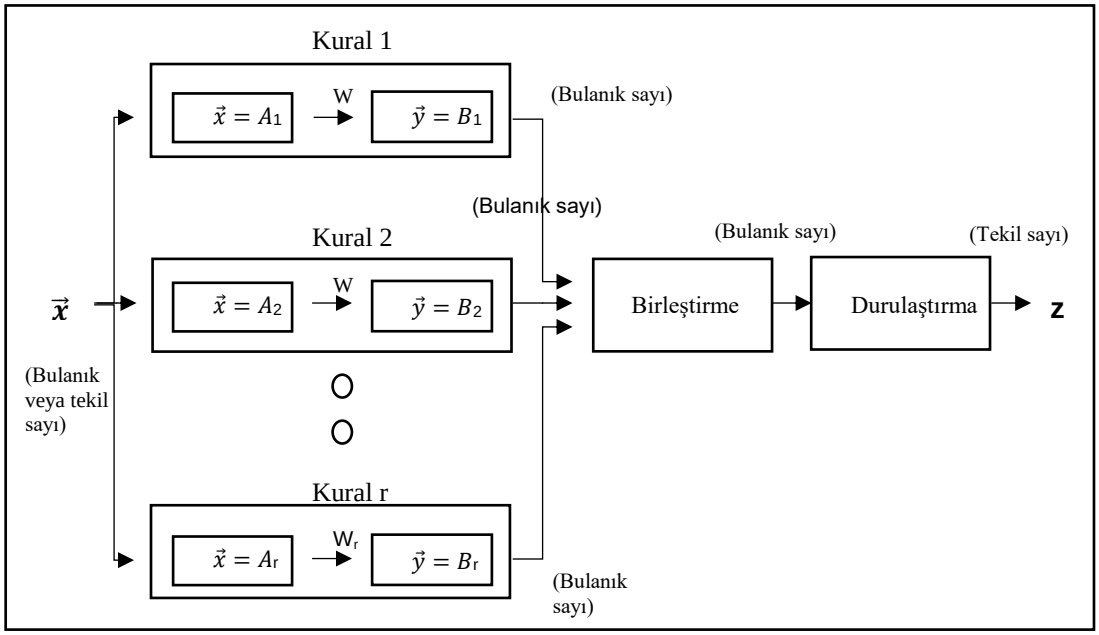
Bulanık mantığa göre ise 30 yaşındaki bir insanın orta yaş grubuna göre bir üyelik derecesi 29 yaşındaki birine göre fazladır. 30 yaşındaki bir insan orta yaş grubuna 0,9 üyelik derecesine sahip iken 29 yaşındaki biri 0,8 derecesine sahip olabilir. Aynı şekilde 20 yaşındaki biri genç yaş grubuna 0,9 üyelik derecesine sahip iken 29 yaşındaki biri 0,2 üyelik derecesinde bu gruba üye olabilir (Şekil 2.1).



Şekil 2.1 Bulanık küme üyelik fonksiyon çizelgesi

2.3 Bulanık Çıkarım Sistemleri

Bulanık Kümelerle elde edilen girdi verileriyle bir karar vermek için bulanık çıkarım sistemleri kullanılmaktadır (Şekil 2.2).



Şekil 2.2 Bulanık çıkarım sistemi blok diyagramı (Jang, Sun ve Mizutani, 1997)

Yukarıdaki şekilde bir bulanık çıkarım sistemi genel olarak gösterilmiştir. $\vec{x}$ bulanık veya tekil girdi sayıları oluşturulan kurallar ile değerlendirilir. Her bir kural için bulanık bir çıktı değeri elde edilir. Elde edilen bu değerler belirlenen bir birleştirme yöntemi kullanılarak tek bir bulanık sayı değerine çevrilir. Bu bulanık değer de belirlenen bir durulaştırma yöntemi ile tekil bir sayıya çevrilir ve **z** çıktı değeri elde edilir.

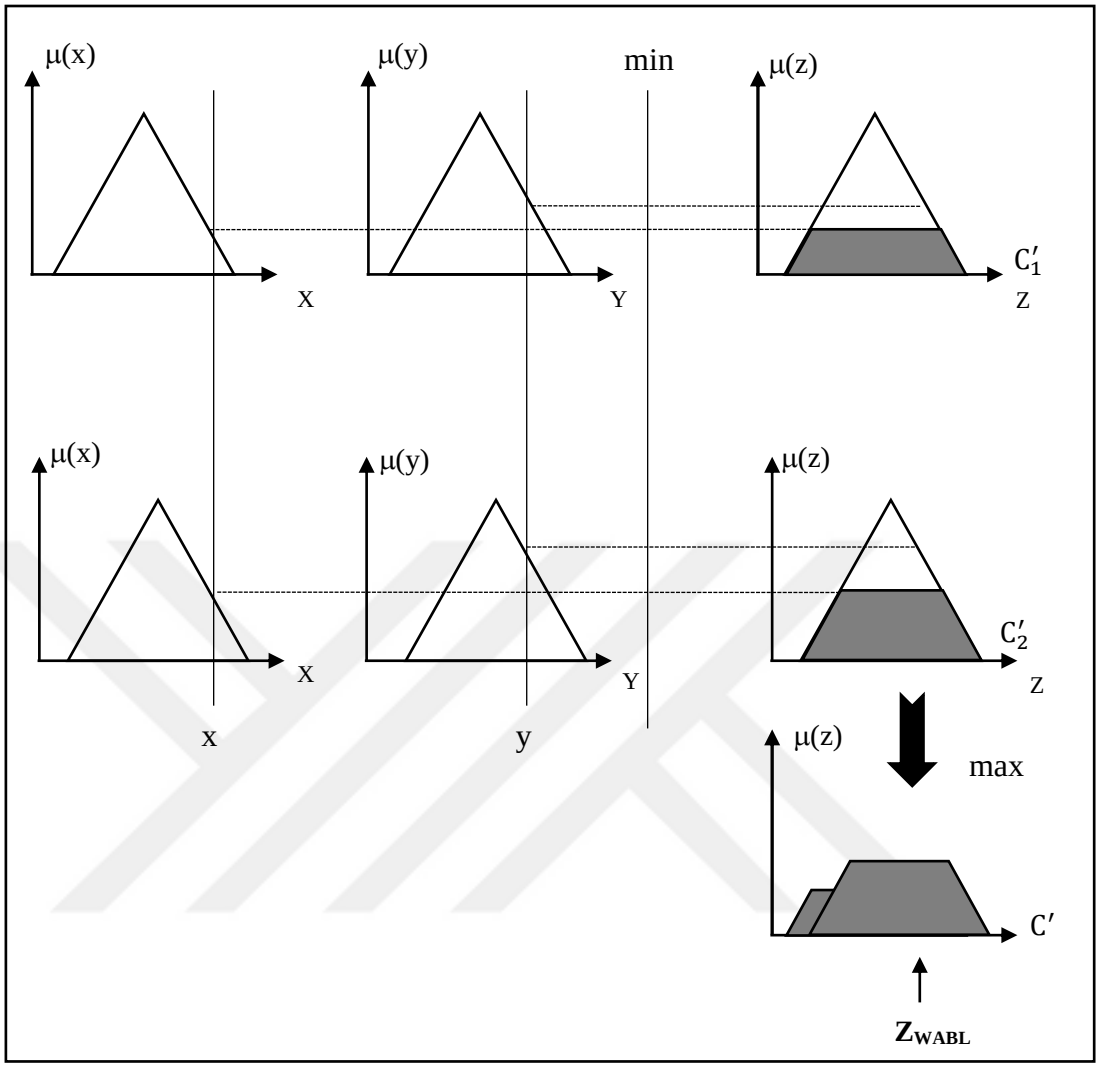
Dört çeşit bulanık çıkarım sistemi mevcuttur. Bunlar; Mamdani Bulanık Çıkarım Sistemi, Larsen Bulanık Çıkarım Sistemi (buna çarpımsal Mamdani Bulanık Çıkarım Sistemi de denilir), Sugeno Bulanık Çıkarım Sistemi ve Tsukamoto Bulanık Çıkarım Sistemidir. Çalışmamızda Mamdani Bulanık Çıkarım Sistemi kullanıldığı için diğer sistemlerin ayrıntılarına girilmeyecektir.

2.3.1 Mamdani Bulanık Çıkarım Sistemi

Mamdani Bulanık Çıkarım Sisteminde girdiler dilsel değişkenler ile belirlenir. Bu dilsel değişkenler için oluşturulan kurallar neticesinde yine bir dilsel değişken olan çıktı değeri elde edilir. Örnek olarak bir evin piyasa değerini belirlenmek istenmektedir. Evin büyüklüğü birinci girdi değeri (X), yaşı ikinci girdi değeri (Y), fiyatı ise çıktı değeri (z) olarak alınsın:

Kural 1: X küçük ve Y yeni ise Z orta,
Kural 2: X küçük ve Y eski ise Z ucuz,
Kural 3: X büyük ve Y yeni ise Z pahalı,
Kural 4: X büyük ve Y eski ise Z orta

Oluşturulan kurallara göre her bir kuralın çıktısı dilsel değişken olarak belirlenir. Girdi değerlerinin X ve Y'ye göre üyelik dereceleri tespit edilir. Her bir kural için Z çıktı değerinin üyelik derecesi X ve Y'nin üyelik derecelerinin en küçüğü olarak bulunur. Diğer tüm kurallar için de aynı şekilde Z'nin üyelik dereceleri bulunur. Belirli bir durulaştırma yöntemi kullanılarak z çıkış tekil değeri elde edilir (Şekil 2.3).



Şekil 2.3 Mamdani bulanık çıkarım sistemi (Jang ve diğer., 1997)

Durulaştırma yöntemi olarak çeşitli yöntemler kullanılabilir. En çok kullanılan durulaştırma yöntemlerinden COA (Center of Area), BOA (Bisector of Area), MOM (Mean of Maxima), WABL (Weighted Average Based on Levels) vs. gösterilebilir. Bu çalışmada WABL durulaştırma yöntemi kullanılmaktadır. Bu bakımdan WABL yöntemi 2.4 bölümünde detaylı şekilde anlatılacaktır.

2.3.2 Sugeno Bulanık Çıkarım Yöntemi

Sugeno Bulanık Çıkarım Yönteminde girdiler; Mamdani Bulanık Çıkarım Sisteminde olduğu gibi dilsel değişkenler ile belirlenir ancak çıktı bir fonksiyondur.

$$Eğer\ x = A\ ve\ y = B\ ise\ z = f(x, y) \quad (2.1)$$

Bu yöntemde her bir kural için fonksiyon sonucunda tekil bir değer elde edilir. Daha sonra ağırlıklı ortalama yöntemi kullanılarak tekil sonuç elde edilir. Mamdani Bulanık Çıkarım Sisteminde uygulanan durulaştırma işlemi burada uygulanmaz. Bir önceki örnek üzerinden gidilirse;

Kural 1: X küçük ve Y yeni ise z = -x+y+1,
Kural 2: X küçük ve Y eski ise z = -y+2,
Kural 3: X büyük ve Y yeni ise z = -x+1,
Kural 4: X büyük ve Y eski ise z = x+y+4

Her bir kuralın üyelik derecesi olarak, giriş değerlerinin üyelik derecelerinin en küçüğü alınır. Mamdani Bulanık Çıkarım Sisteminde olduğu gibi; her bir kural için z çıktı değerinin üyelik derecesi X ve Y'nin üyelik derecelerinin en küçüğü olarak bulunur (Şekil 2.4).

Belirlenen fonksiyonlardan z_1 ve z_2 değerleri, elde edilir. Minimum üyelik dereceleri olan w_1 ve w_2 tespit edilir ve ağırlıklı ortalama yöntemi kullanılarak z değeri bulunur.

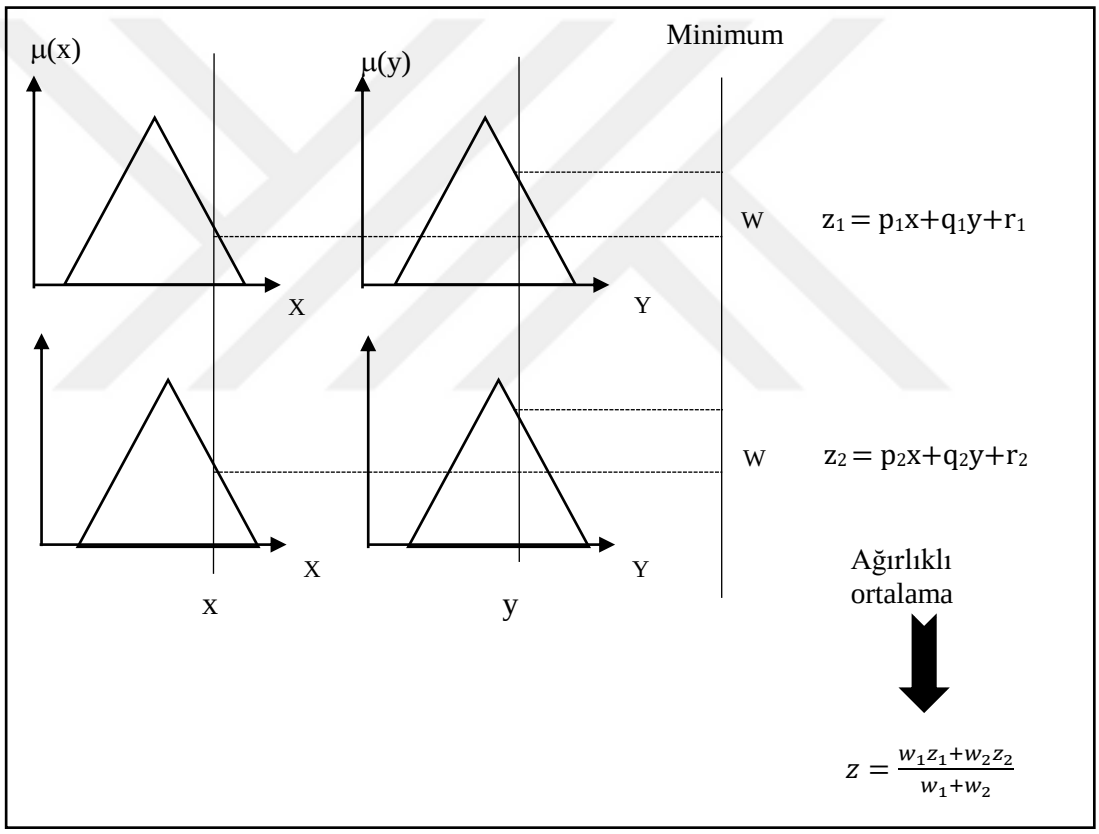
2.3.3 Tsukamoto Bulanık Çıkarım Sistemi

Tsukamoto Bulanık Çıkarım Sisteminde diğer yöntemlerde olduğu gibi; girdiler dilsel değişkenler ile belirlenir. Bu dilsel değişkenler için oluşturulan kurallar neticesinde monotonik bir fonksiyon olan çıktı değeri elde edilir. Yukarıdaki örnek üzerinden incelendiğinde kurallar şu şekilde belirlenir:

Kural 1: X küçük ve Y yeni ise Z = C_1,
Kural 2: X küçük ve Y eski ise Z = C_2,
Kural 3: X büyük ve Y yeni ise Z = C_3,
Kural 4: X büyük ve Y eski ise Z = C_4

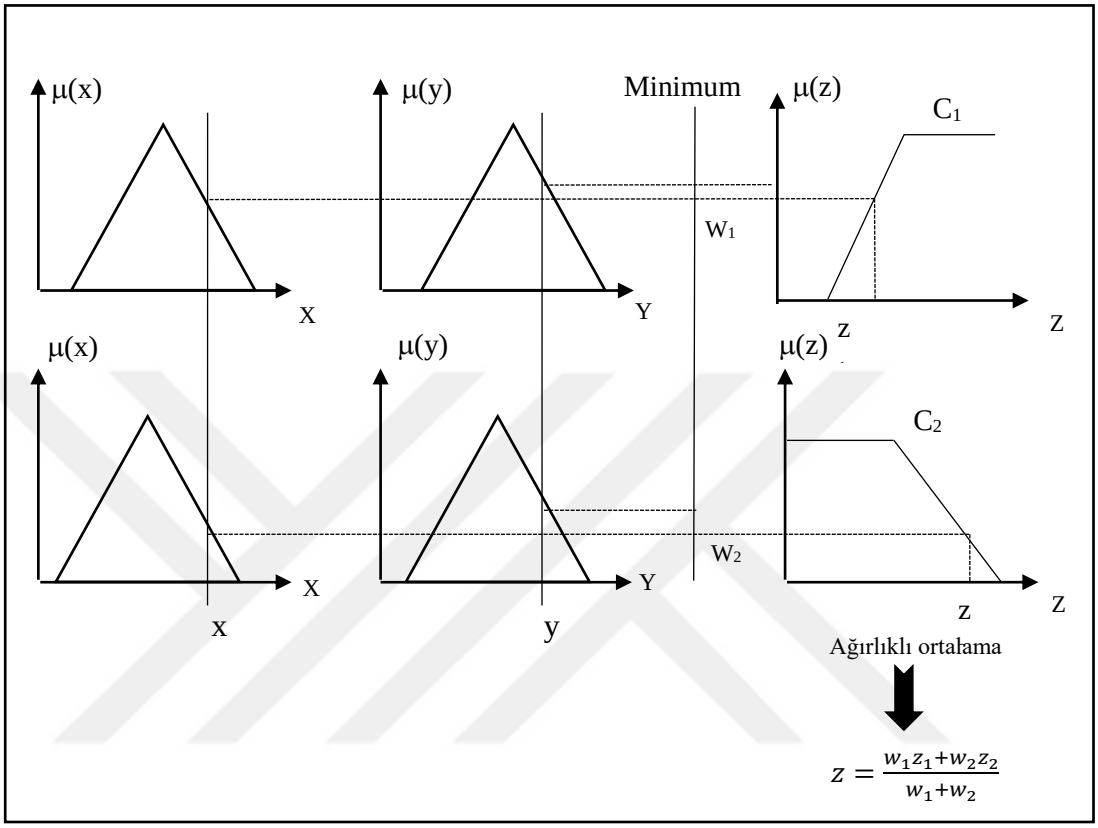
Her bir kuralın üyelik derecesi olarak, giriş değerlerinin üyelik derecelerinin en küçüğü alınır. Diğer Bulanık Çıkarım Sistemlerinde olduğu gibi; her bir kural için z çıktı değerinin üyelik derecesi X ve Y'nin üyelik derecelerinin en küçüğü olarak bulunur.

Bu yöntemde de Sugeno Bulanık Çıkarım Sistemi gibi her bir kural için tekil bir değer elde edilir. Daha sonra ağırlıklı ortalama yöntemi kullanılarak tekil sonuç elde edilir. Mamdani Bulanık Çıkarım Sisteminde uygulanan durulaştırma işlemi burada da uygulanmaz.



Şekil 2.4 Sugeno bulanık çıkarım sistemi (Jang ve diğer., 1997)

Yukarıdaki örnekte gösterildiği gibi; z_1 ve z_2 değerleri, belirlenen fonksiyonlardan elde edilir. Minimum üyelik dereceleri olan w_1 ve w_2 tespit edilir ve ağırlıklı ortalama yöntemi kullanılarak z değeri bulunur (Şekil 2.5).



Şekil 2.5 Tsukamoto bulanık çıkarım sistemi (Jang ve diğer., 1997)

2.4 WABL Yöntemi

Bulanık sayıların durulaştırılmasında kullanılan WABL yönteminin amacı, bulanık sayının ağırlık merkezini seviyelere göre bulmaktır. Sürekli bir bulanık sayı olan A'nın WABL durulaştırma yöntemine göre değeri aşağıdaki formül ile hesaplanmaktadır:

$$WABL(A) = \int_0^1 (cR_A(\alpha) + (1-c)L_A(\alpha))p(\alpha)d\alpha \qquad (2.2)$$

Formülde yer alan $c \in [0,1]$, karar vericinin iyimserlik katsayısını, $p(\alpha)$ fonksiyonu ise bulanık sayının seviye kümelerinin önem ağırlıklarının dağılım fonksiyonudur ve aşağıdaki koşulları sağlamalıdır:

$$\int_0^1 p(\alpha)d\alpha = 1 \tag{2.3}$$

$$p(\alpha) \geq 0, \ \forall \alpha \in (0,1] \tag{2.4}$$

Literatürde $p(\alpha)$ fonksiyonunun aşağıdaki gibi parametrik tanımı kullanılmaktadır (Nasibov ve Mert, 2007; Nasiboğlu ve Abdullayeva, 2018):

$$p(\alpha) = (k+1)\alpha^k, \ k = 0,1,2,\ldots \tag{2.5}$$

(2.5) şeklinde belirlenen $p(\alpha)$ fonksiyonu durumunda bulanık sayıların WABL değerlerini kolay şekilde hesaplamak için çeşitli çalışmalar mevcuttur. (Nasiboğlu ve Abdullayeva, 2018; Nasibov ve Mert, 2007). Örneğin; Nasibov ve Mert (2007) çalışmasında aşağıdaki teorem ispatlanmıştır:

Teorem 1: Bulanık yamuk $A(l, m_l, m_r, r)$ sayısının (Şekil 2.6) seviye ağırlık fonksiyonu (2.5) şeklinde olduğu durumda, WABL değeri aşağıdaki gibi hesaplanabilir:

$$WABL(A) = c\left(r - \frac{k+1}{k+2}(r - m_r)\right) + (1-c)\left(l + \frac{k+1}{k+2}(m_l - l)\right) \tag{2.6}$$

Nasiboğlu ve Abdullayeva (2018) çalışmasında ise eşit ağırlıklı seviye kümeleri (k=0) halinde aşağıdaki teorem ispatlanmıştır:

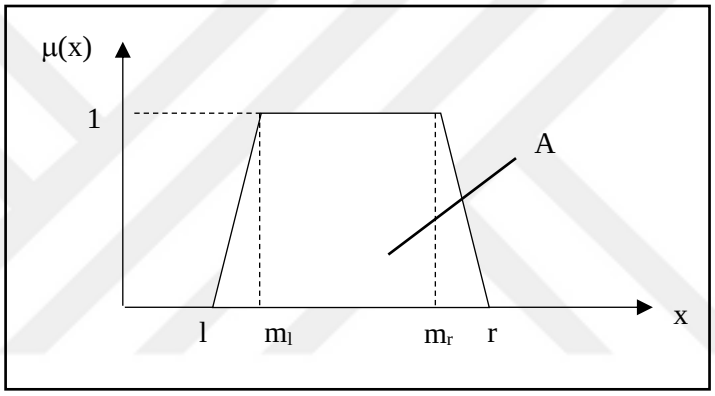
Teorem 2: Bulanık yamuk $A(l, m_l, m_r, r)$ sayısının seviye ağırlıkları sabit dağıldığı durumda (k=0) , WABL değeri aşağıdaki gibi hesaplanabilir:

$$WABL(A) = \frac{M(0)+M(1)}{2} \tag{2.7}$$

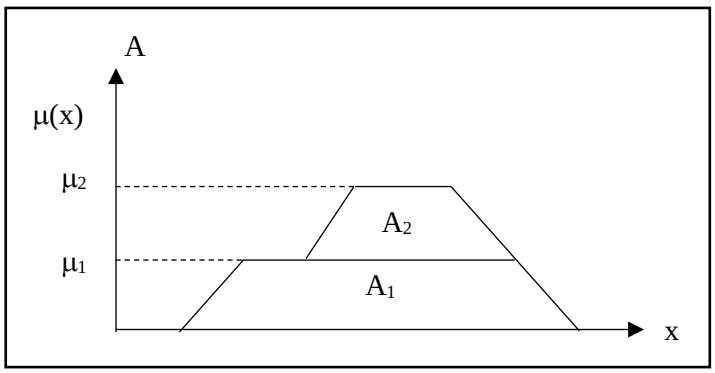
$$M(0) = (1-c)l + cr \tag{2.8}$$

$$M(1) = (1 - c)m_l + cm_r \tag{2.9}$$

Fakat yukarıdaki teoremlerden de görüldüğü gibi, (Nasibov ve Mert, 2007; Nasiboğlu ve Abdullayeva, 2018) çalışmalarında WABL yönteminin hesaplama formülleri sadece üçgen ve yamuk şeklinde bulanık sayılar için geçerlidir. Bu tez çalışmasında bulanık çıkarsama sistemi sonucu oluşan bulanık sayılar ise üçgen ve yamuk sayıların kompozisyonu şeklinde gösterilebilecek karmaşık şekilli bulanık sayılardan oluşmaktadır (Şekil 2.7). Bu bakımdan, yukarıdaki teorem 1 ve teorem 2'de önerilen analitik formulalar direkt olarak kullanılamamaktadır. Bu durum dikkate alınarak, tez çalışmasında yeni teorem ispatlanmıştır.



Şekil 2.6 Yamuk üyelik fonksiyonuna sahip bulanık sayı (Nasiboğlu ve Abdullayeva, 2018)



Şekil 2.7 İki alana bölünmüş kompozit bulanık sayı grafiği

Teorem 3: Bulanık yamuk sayıların kompozisyonundan oluşan karmaşık sayının (Şekil 2.7) seviye ağırlıkları sabit dağıldığı durumlarda (k=0) , WABL değeri aşağıdaki gibi hesaplanabilir:

$$WABL(A) = \mu_1 WABL(A_1) + (\mu_2 - \mu_1)WABL(A_2) \tag{2.10}$$

İspatı: WABL değerinin genel olarak aşağıdaki gibi hesaplandığını hatırlayalım:

$$WABL(A) = \int_0^1 (cR_A(\alpha) + (1-c)L_A(\alpha))p(\alpha)d\alpha \quad (2.11)$$

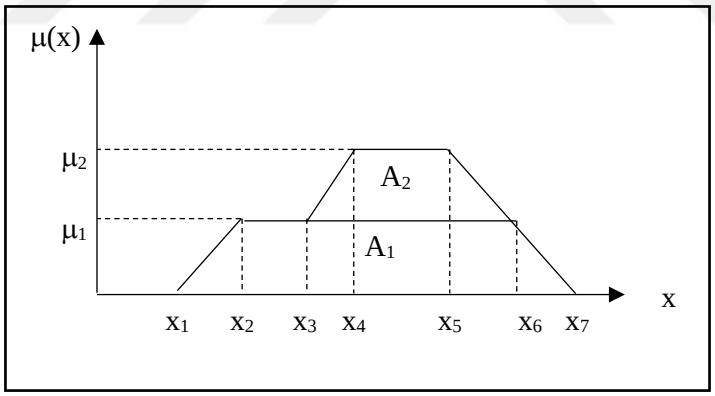
$$\int_0^1 p(\alpha)d\alpha = 1, \quad p(\alpha) \geq 0, \ \forall \alpha \in (0,1] \quad (2.12)$$

$$R_A(\alpha) = r - \alpha(r - m_r) \quad (2.13)$$

$$L_A(\alpha) = l + \alpha(m_l - l) \quad (2.14)$$

Yukarıdaki (2.13) ve (2.14) formüllerinde olan $l,\ r,\ m_l$ ve m_r değerlerinin yerine Şekil 2.8'deki uygun x değerlerini koyarsak ve integrali $[0, \mu_1]$ ve $[\mu_1, \mu_2]$ aralıklarına ayırırsak, aşağıdakileri yazabiliriz:

$$WABL(A) = \int_0^{\mu_1} \left[c\left(x_7 - \alpha(x_7 - x_6)\right) + (1-c)\left(x_1 + \alpha(x_2 - x_1)\right)\right]p(\alpha)d\alpha +$$
$$+ \int_{\mu_1}^{\mu_2} \left[c\left(x_6 - \alpha(x_6 - x_5)\right) + (1-c)\left(x_3 + \alpha(x_4 - x_3)\right)\right]p(\alpha)d\alpha \quad (2.15)$$



Şekil 2.8 A1=(x1,x2,x6,x7) ve A2=(x3,x4,x5,x6) bulanık sayıları grafiği

(2.15) sadeleştirildiği zaman:

$$WABL(A) = \mu_1[c(x_6 - x_7) + (1-c)(x_2 - x_1)] +$$
$$+(\mu_2 - \mu_1)[c(x_5 - x_6) + (1-c)(x_4 - x_3)] \quad (2.16)$$

elde edilir. A_1 ve A_2 yamuk bulanık sayılarının (Şekil 2.8) WABL değerleri ayrı ayrı alınırsa:

$$WABL(A_1) = \int_0^1 \left[c\left(x_7 - \alpha(x_7 - x_6)\right) + (1-c)\left(x_1 + \alpha(x_2 - x_1)\right)\right] p(\alpha) d\alpha \quad (2.17)$$

$$WABL(A_1) = \mathrm{c}(x_6 - x_7) + (1-c)(x_2 - x_1) \quad (2.18)$$

$$WABL(A_2) = \int_0^1 \left[c\left(x_6 - \alpha(x_6 - x_5)\right) + (1-c)\left(x_3 + \alpha(x_4 - x_3)\right)\right] p(\alpha) d\alpha \quad (2.19)$$

$$WABL(A_2) = \mathrm{c}(x_5 - x_6) + (1-c)(x_4 - x_3) \quad (2.20)$$

elde edilir. (2.18) ve (2.20) eşitliklerini (2.16) formulünde dikkate alırsak,

$$WABL(A) = \mu_1 WABL(A_1) + (\mu_2 - \mu_1) WABL(A_2) \quad (2.21)$$

olduğu görünür. Bununla teoremin ispatı tamamlanmış oldu.

BÖLÜM ÜÇ
K-ORTALAMALAR ALGORİTMASI

3.1 Tanımı

K-Ortalamalar Algoritmasının amacı; bir grup verinin (elemanın) belirlenen sayıdaki küme içerisinde hangi kümeye ait olduğunu tespit etmektir. K-Ortalamalar Algoritması 5 adımdan oluşmaktadır;

Adım 1: Kaç küme olacağı kullanıcı tarafından belirlenir.

Adım 2: Küme merkezleri rastgele değer atanarak belirlenir.

Adım 3: Her bir eleman için her küme merkezine olan mesafe tespit edilir ve eleman en yakın merkezli kümeye atanır.

Adım 4: Oluşturulan her küme için yeniden küme merkezi hesaplanır.

Adım 5: Küme merkezindeki yer değişimi durana kadar adım 3'e gidilir.

Adım 3'te belirtilen mesafenin tespiti için genel olarak Öklid Mesafesi (Euclidean Distance) kullanılmaktadır. Küme merkezlerinde bir değişim olmadığı zaman algoritma sonlandırılır. Algoritmayı sona erdirmenin diğer bir yolu ise yakınsama kriterinin sağlanmış olmasıdır. Hata kareler yöntemi ile hesaplanan değerler arasında önemli bir değişim olmadığı zaman algoritma sonlandırılabilir. $p \epsilon C_i$ i kümesindeki her bir veri noktasını, m_i ise i kümesinin merkez noktasını temsil etmektedir.

$$dist\big((x,y),(a,b)\big) = \sqrt{(x-a)^2 + (y-b)^2} \tag{3.1}$$

$$SSE = \sum_{i=1}^{k} \sum_{p \epsilon C_i} d(p, m_i)^2 \tag{3.2}$$

3.2 Örnek Uygulama

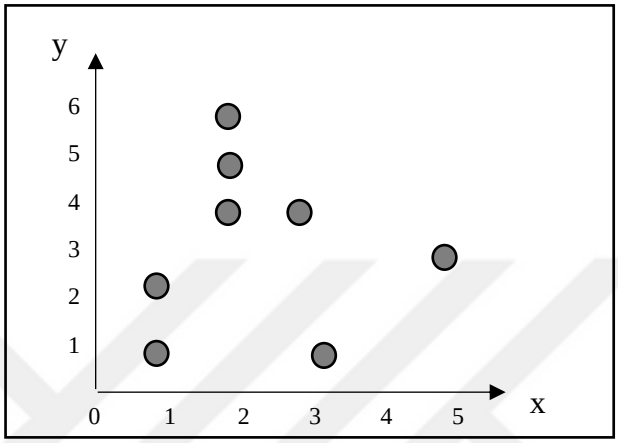
8 noktadan oluşan bir koordinat grubu oluşturalım ve K-ortalama algoritmasının adım adım uygulayalım (Şekil 3.1);

a = (1, 1), b = (2, 4), c = (3, 4), d = (1, 2), e = (3, 1), f = (5, 3), g = (2, 6), h = (2, 5)

Adım 1: Küme sayısını k=2 olarak atayalım.

Adım 2: k sayısı kadar rastgele iki noktayı merkezi noktalar olarak atayalım.

$$m_1 = (1, 1) , m_2 = (2, 2)$$



Şekil 3.1 Küme çizelgesi

Adım 3: Her bir nokta için en yakın küme merkezini bulalım. Öklid mesafe ölçümlerine göre, noktaların küme merkezlerinden uzaklıkları Tablo 3.1'de hesaplanmıştır;

Tablo 3.1 Noktaların küme merkezlerinden uzaklıkları (başlangıç durum)

	a	**b**	**c**	**d**	**e**	**f**	**g**	**h**
m_1'e uzaklığı	0	3,16	3,61	1	2	4,47	5,1	4,12
m_2'ye uzaklığı	1,41	2	2,24	1	1,41	3,16	4	3
Kümelere atama	**C_1**	**C_2**	**C_2**	**C_1**	**C_2**	**C_2**	**C_2**	**C_2**

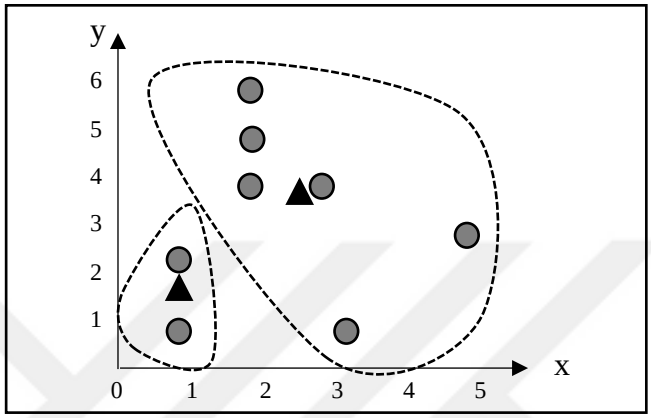
Tablo 3.1'de görüldüğü üzere, $a,d \in C_1$; $b,c,e,f,g,h \in C_2$ olarak tespit edilir. Hata kareler toplamı:

$$SSE = 0^2 + 2^2 + 2{,}24^2 + 1^2 + 1{,}41^2 + 3{,}16^2 + 4^2 + 3^2 = 46{,}99$$

Adim 4: Her bir küme için merkezi noktayı tekrar bulalım. Her bir kümeye ait olan koordinatların aritmetik ortalaması alınarak yeni merkezi noktalar bulunur (Şekil 3.2):

$$m_1 = ((1+1)/2, (1+2)/2) = (1,(1,5))$$

$$m_2 = ((2+3+3+5+2+2)/6, (4+4+1+3+6+5)/6) = ((2,83),(3,83))$$



Şekil 3.2 İlk iterasyon sonucu küme çizelgesi

Adım 5: Küme merkezlerinin yeri değiştiğinden dolayı adım 3'e gidilir.

Adım 3: İlk iterasyon sonucu, noktaların yeni küme merkezlerinden hesaplanmış uzaklıkları Tablo 3.2'de verilmiştir;

Tablo 3.2 Noktaların yeni küme merkezlerinden uzaklıkları (İlk iterasyon sonucu)

	a	**b**	**c**	**d**	**e**	**f**	**g**	**h**
m_1'e uzaklığı	0,5	1,12	3,2	0,5	2,06	4,27	4,6	3,64
m_2'ye uzaklığı	3,37	2	2,59	2,59	3,37	2,32	2,32	1,43
Kümelere atama	**C_1**	**C_1**	**C_2**	**C_1**	**C_1**	**C_2**	**C_2**	**C_2**

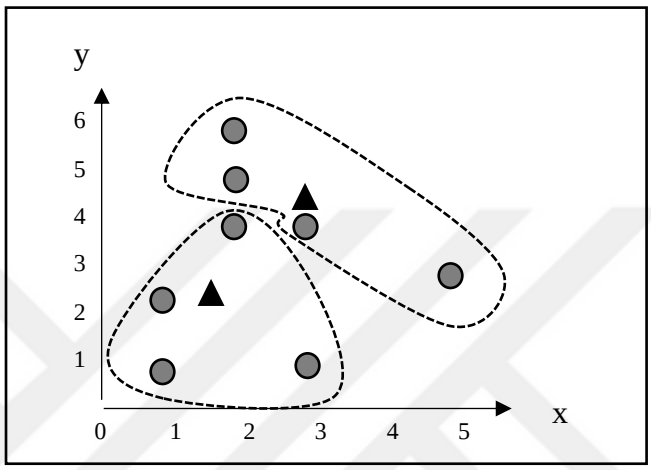
Tablo 3.2'de görüldüğü üzere, ilk iterasyon sonrası a,b,d,e $\in C_1$; c,f,g,h $\in C_2$ olarak tespit edilir. Hata kareler toplamı ise:

$$SSE = 0,5^2 + 1,12^2 + 2,59^2 + 0,5^2 + 2,06^2 + 2,32^2 + 2,32^2 + 1,43^2 = 25,52$$

Adim 4: Her bir küme için merkezi noktayı tekrar bulalım (Şekil 3.3).

$$m_1 = ((1+2+1+3)/4, (1+4+2+1)/4) = ((1,75), 2)$$
$$m_2 = ((3+5+2+2)/4, (4+3+6+5)/4) = (3, (4,5))$$

Adım 5: Küme merkezindeki değişim durana kadar adım 3'e gidilir.

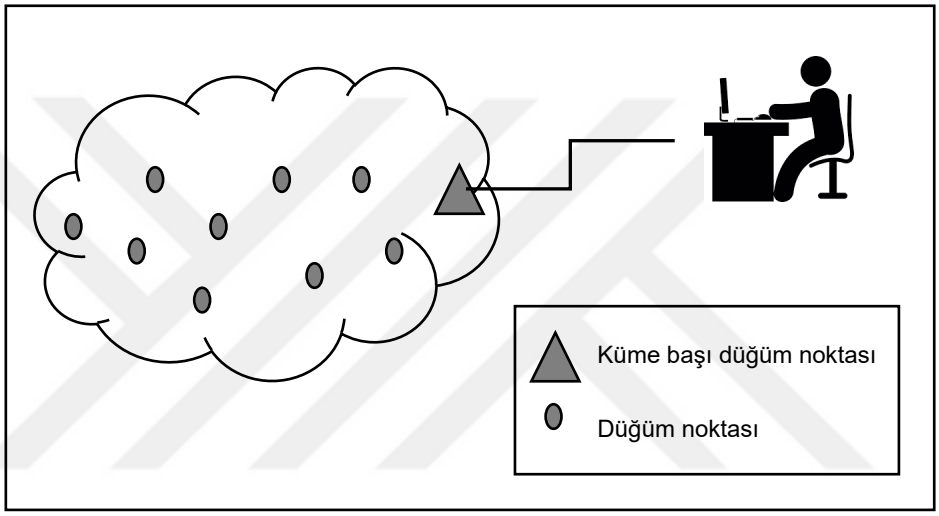


Şekil 3.3 İkinci iterasyon sonrası küme çizelgesi

BÖLÜM DÖRT
KABLOSUZ SENSÖR AĞLARI

4.1 Tanımı

Kablosuz sensör ağları, kablosuz ortamda birden fazla düğüm noktasından oluşan bir iletişim ağıdır. Düğümlerin sayısı ve topolojisi değişkendir. Bu sebeple istenilen ölçüde bir ağ kolaylıkla oluşturulabilir (Şekil 4.1).



Şekil 4.1 Kablosuz sensör ağı

Oluşturulan kablosuz ağ sayesinde istenilen verilere kolaylıkla ulaşılabilir. Kullanıcılar tarafından işlenen ve çözümlenen veriler bir araya getirilerek bilgi niteliği taşırlar. Kablosuz sensör ağları geniş bir yelpazede birçok alanda kullanılmaktadır. Bunun nedeni:

- Güvenilirlik
- Doğruluk
- Esneklik
- Maliyet verimliliği
- Kurulum kolaylığı özelliklerinin bulunmasıdır.

Düğüm noktalarının ömrü pillerinin ömrüne göre belirlenir. Düğüm noktasının fazlalığı, düğümler arasındaki mesafe ve gönderilen verinin büyüklüğü bu ömrün başlıca belirleyici parametreleridir.

4.2 Kablosuz Sensör Ağlarının Kullanımı

Kablosuz sensör ağları; kurulumu kolay ve esnek bir yapıda olduğu için birçok alanda tercih edilmektedir. Kablolu ağların kurulumunun imkânsız olduğu durumlarda tercih edilmekle birlikte tüm çevresel koşullara uyum sağlayabilmektedir. Kablosuz sensör ağlarının başlıca kullanım alanları:

- Bilimsel Araştırmalar
- Sağlık Uygulamaları
- Askeri Uygulamalar
- Çevrenin İzlenmesi ve Korunmasına Yönelik Uygulamalar
- Ev Uygulamaları
- Reklam Uygulamaları
- Arazi Gözlemleri
- Hava Kirliliği Gözlemleri
- Orman Yangını Tespiti
- Toprak Kayması Tespiti
- Su Kalitesi Gözlemleri
- Endüstriyel Gözlemler
- Güvenlik Hizmetleri
- Ulaşım Hizmetleri olarak sıralanabilir.

Daha belirgin örnekler verilecek olursa:

- Bir orman yangınında uçaktan sensörler atılarak bir sıcaklık haritası çıkartılabilmektedir.
- Akıllı bina uygulamalarıyla sıcaklık ve nem ölçümü ile enerji tasarrufu sağlanabilmekte; odaların doluluğu, sıcaklığı vs tespit edilebilmektedir.
- Vahşi yaşam alanlarına yerleştirilen sensörler ile o bölgede yaşayan canlılar ile ilgili gözlem yapılabilmektedir.

- Araç çarpışma sistemleri, lastik basınç sensörleri gibi sistemler ile daha güvenli yolculuk yapılabilmektedir.
- Askeri operasyonlarda birliklerin bulundukları koordinatlar, ilerleme yönleri, hızları vs. tespit edilebilmekte ve gözlenebilmektedir.
- Ulaşım firmaları araçlarında bulunan sensörler vasıtasıyla koordinatlarını, hızlarını, ağırlıklarını takip edebilmektedir.
- Optik sensörler vasıtasıyla çiftçiler, ürünlerinin aldığı güneş enerjisi miktarını gözlemleyebilmektedir. Seraların sıcaklığı, nemi, güneş ışığına maruz kalma oranı, oksijen seviyesi takip edilebilmektedir.

4.3 Kablosuz Sensör Ağının Gereksinimleri

Sensör ağlarının gereksinimleri şu şekilde sıralanabilir:

- Kullanım Ömrü: Kullanılan sensörlere daha sonradan erişmenin imkânsız olduğu durumlar olabileceği için ağın kullanım ömrünün maksimum olması hedeflenir.
- Ağın büyüklüğü: Çoğu uygulamalarda daha büyük coğrafyaya yayılan sensörler daha fazla bilgi alınmasını sağlayabilir. Ağın büyüklüğü uygulamanın amacına ve istenilen verinin miktarına göre değişebilir.
- Hatayı en aza indirgeme: Ağ yapısının hatalı olması, sensörlerin yetersizliği verilerin hatalı veya eksik gönderilmesine sebep olur. Bu durum da toplanılan verinin güvenilir ve işlenebilecek düzeyde olmamasına neden olur. Sonuç olarak; ağ ve sensörler tasarlanırken hata oranlarına dikkat edilmesi gerekir.
- Düşük enerji tüketimi: Sensörlerin enerji sarfiyatı ağın kullanım ömrünü etkileyeceğinden dolayı, sensörler ile baz istasyonu arasındaki enerji kullanımının minimum olması hedeflenir.
- Çoklu atlama (multi-hop) iletişime uyumlu olması: Sensörlerin direkt olarak baz istasyonu ile iletişim halinde olmaları, enerji sarfiyatının mesafeyle doğru orantılı olduğu göz önünde alındığında, ağın kullanım ömrünü büyük ölçüde azaltacaktır. Bu sebeple; sensörlerin diğer sensörleri aracı kullanarak iletişimi sağlamak daha avantajlıdır.

- Ölçeklenebililirlik: Sensörler arası bağlantının kurulumu ve idamesinden kullanılan iletişim protokolü güvenilir olmalıdır. Ağ büyüdükçe kullanılan protokolün performansı ve güvenilirliğinde bir değişme olmamalıdır.
- Güvenilirlik: Güvenilir veri iletimi, bilginin eksiksiz ve doğru tanımlanmasında büyük rol oynamaktadır. İletim esnasındaki paket kayıpları sistemi etkisiz kılabilmektedir.

4.4 Kablosuz Sensör Ağlarında Güvenlik

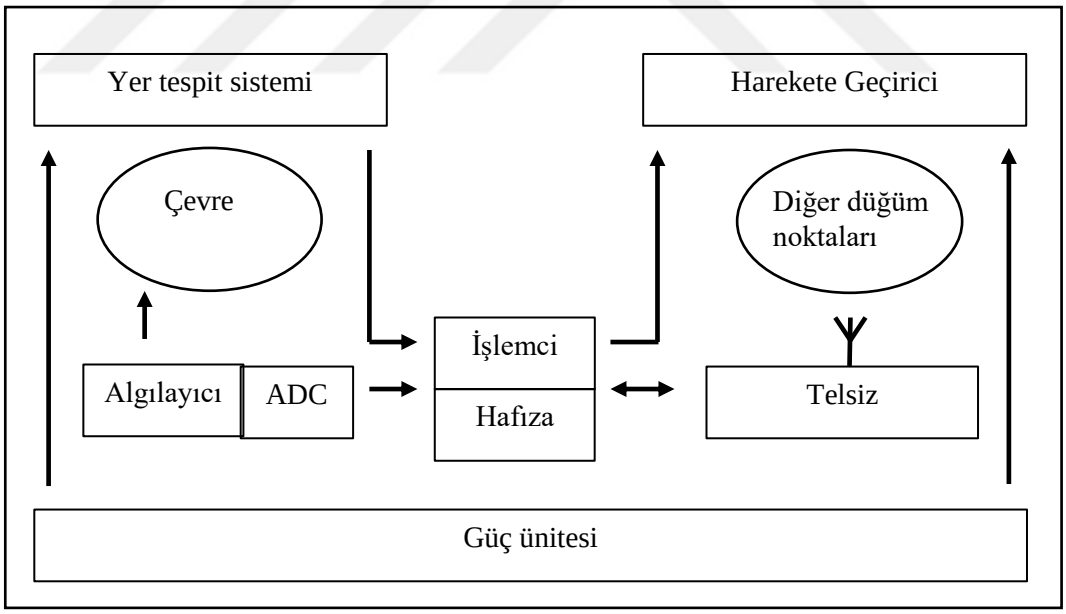
Kablosuz sensör ağları her türlü siber saldırıya karşı hassastır. Güvenlik uygulamalarında aşağıdaki hususlara dikkat edilmelidir:

- Gizlilik: Bilginin yetkisiz kişilerden korunmasıdır. Bilginin saklanması ve iletilmesi esnasında kriptolama yapılması esasına dayanmaktadır.
- Kullanılabilirlik: Bilginin her ihtiyaç duyulduğunda yetkili kişiler tarafından kullanılabilmesidir. Kullanıcılar her an güncel veriye ulaşabilmeli, yedekleme altyapısının yeterli olması sağlanmalıdır.
- Bütünlük: Bilginin yetkisiz kişiler tarafından silinmesine veya değiştirilmesine karşı yapılan korumadır. Yetkilendirilmiş kişilerin bilgiye erişimini sağlamak, kullanıcılar arasında yetki sınırlaması yapmak, parola kullanmak başlıca koruma yöntemlerindendir.
- Doğrulama: Düğüm noktalarının kendi aralarında kimliklerini paylaşması esasına dayanır. Kimliği tanınmayan düğüm noktasının veri aktarımı yapmasına veya veri toplamasına engel olur.
- İnkâr-edilemez olması: bir kaynağın gönderdiği veriyi inkar edememesi esasına dayanır. Her verinin hangi kaynaktan gönderildiğinin kayıt altına alınmasıdır.
- Yetkilendirme: Sadece yetkili düğüm noktaların ağın kaynaklarına ve hizmetlerine erişebilmesi esasına dayanır.
- Tazelik: bir düğüm noktasının daha önceden topladığı veriyi tekrar göndermesini engellemeye dayanır. Düğüm noktaları daima güncel verileri gönderir.

- İleriye yönelik gizlilik: Sistemden ayrılan düğüm noktasının geleceğe ait iletileri tahmin edememesi esasına dayanır.
- Geçmişe yönelik gizlilik: Sisteme yeni giren düğüm noktalarının geçmişteki iletileri bilmemeleri esasına dayanır.

4.5 Kablosuz Sensör Ağlarının Yapısı ve Enerji Kullanımı

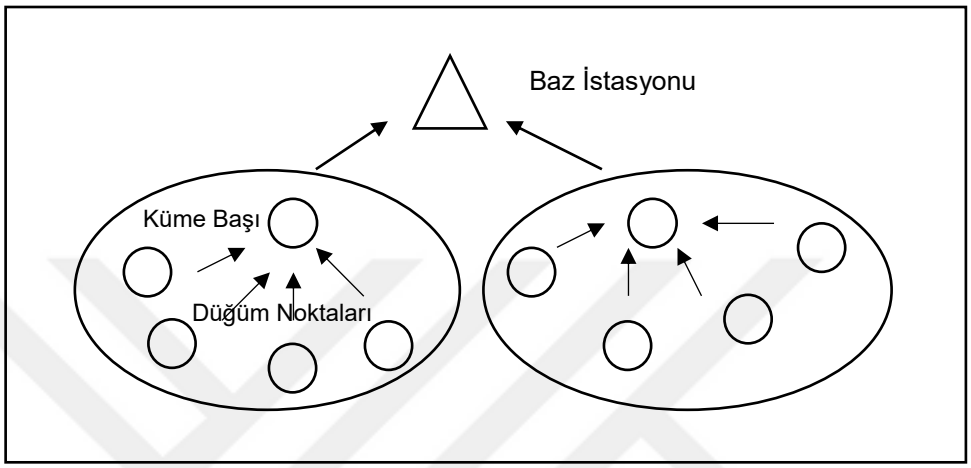
Bir kablosuz ağ sensörü; algılayıcı, işlemci, telsiz ve güç ünitesinden oluşmaktadır. Ayrıca; yer tespit sistemi, güç jeneratörü ve harekete geçirici (mobilizer) gibi ek uygulamaları da bulunabilmektedir. Çevre ile iletişim halinde bulunan sensörler gözlemledikleri fiziksel parametreleri ADC (Analog to Digital Converter) vasıtasıyla sayısal verilere dönüştürür ve işlemciye aktarır. İşlemci veri akışını kontrol eder ve aldığı veriyi telsiz ünitesine iletir. Telsiz ünitesi düğüm noktasını ağa bağlayan çıkış noktasıdır. Kablosuz ağ sensörlerinde, güneş enerjisi gibi çeşitli güç üniteleri kullanılabilmektedir (Şekil 4.2).



Şekil 4.2 Düğüm noktası yapısı (Nithya ve Gomathy, 2015)

Kablosuz sensör ağları çeşitli topoloji sistemleri kullanabilmektedir. Çalışmamızda hiyerarşi modeli kullanılacağı için bu topoloji üzerinden veri aktarımı ve enerji kullanımı anlatılacaktır (Şekil 4.3).

Hiyerarşi modelinde en alt katmanda düğüm noktaları bulunur. Düğüm noktaları bir veya birden fazla kümelere ayrılabilir. Her bir küme elde ettiği verileri küme başı düğüm noktası olarak tabir edilen bir üst katmandaki düğüm noktasına gönderir. Küme başı düğüm noktaları ise topladığı verileri baz istasyonu olarak tabir edilen bir üst noktaya ileterek görevini yerine getirir.



Şekil 4.3 Kablosuz sensör ağı hiyerarşi topolojisi

Küme başlarına düğüm noktalarının m bit kadar veriyi gönderebilmek için harcadıkları enerji miktarı aşağıdaki formül ile hesaplanır (Mehra ve diğer., 2018):

$$E_{Tx}(m,d) = \begin{cases} mE_{elec} + me_{fs}d^2, & d < d_0 \\ mE_{elec} + me_{mp}d^4, & d \geq d_0 \end{cases} \quad (4.1)$$

$$d_0 = \sqrt{\frac{e_{fs}}{e_{mp}}} \quad (4.2)$$

Burada:

E_{elec} : enerji dağılımı (nJ/bit)

e_{fs} : verici amplifikatörü boş alan modeli (pJ/bit/m2)

e_{mp} : verici amplifikatörü çok yollu modeli (pJ/bit/m4)

d_0 : mesafe alt sınır değeri (m)

m : veri paketi uzunluğu (bits)

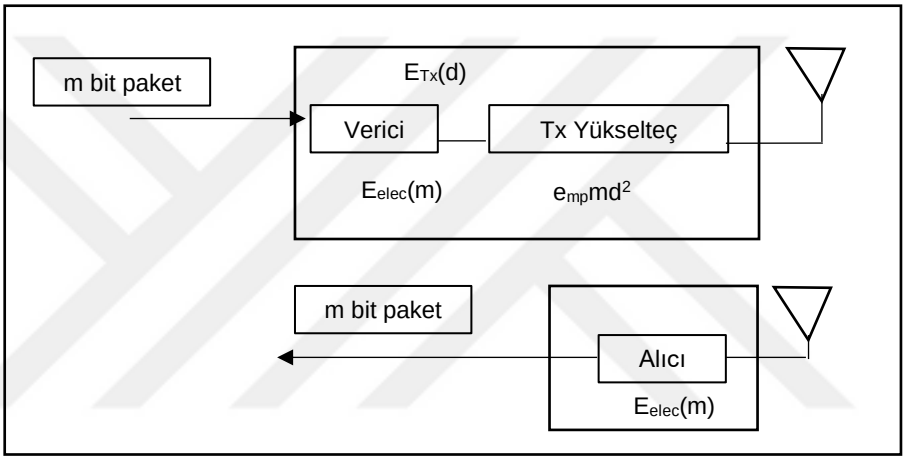
d_{BS} : merkez istasyona olan uzaklık (m)

E_{DA} : veri füzyonunda harcanan enerji (nJ/bit/report)

Aşağıdaki şekilde alıcı ile vericinin telsiz modeli basitçe gösterilmiştir (Şekil 4.4). Enerji formüllerindeki d_0, mesafe eşik değeri olarak tanımlanmaktadır. Bir küme başının n düğüm noktasından aldığı m bit kadar veriyi merkez istasyona gönderebilmek için harcadığı enerji ise (Mehra ve diğer., 2018):

$$E_{Tx}(m,d) = nm(E_{elec} + e_{fs}d_{bs} + E_{DA}) \tag{4.3}$$

olarak hesaplanmaktadır.

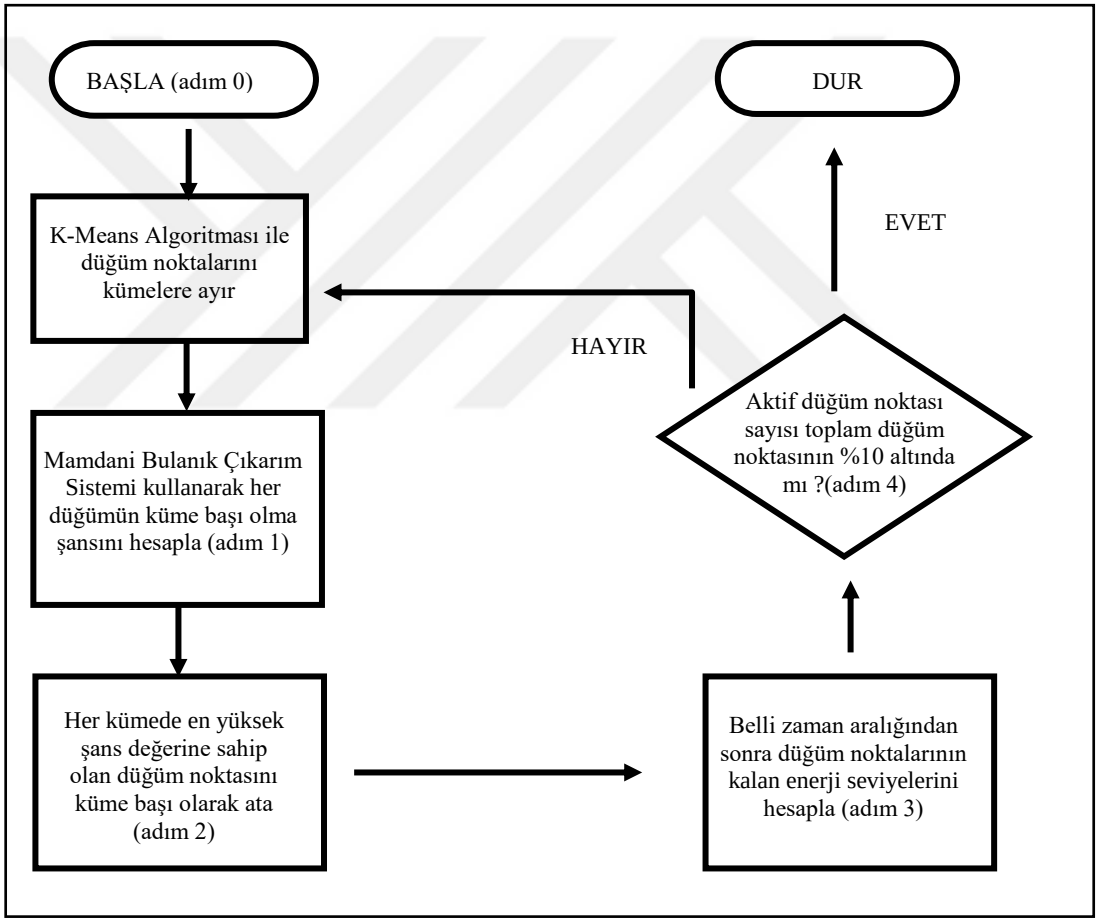


Şekil 4.4 Telsiz modeli (Bidaki ve Tabbakh, 2016)

BÖLÜM BEŞ
UYGULAMA

5.1 Giriş

Uygulamada oluşturulan sistemde belli aralıklarla düğüm noktaları üyesi oldukları küme başı düğüm noktasına veri göndermektedir. Her veri gönderiminde küme başı noktaları bulanık çıkarım sistemine göre yenilenmektedir. Uygulamamızın akış şeması Şekil 5.1'de görüldüğü gibidir.



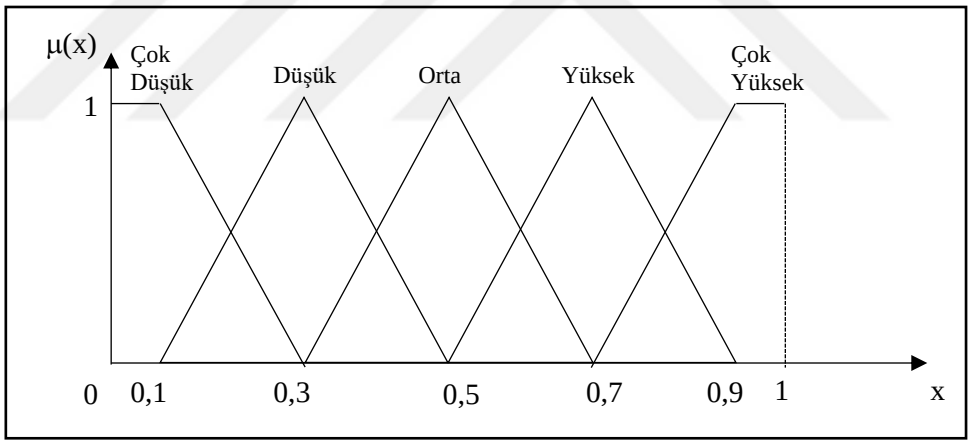
Şekil 5.1 Kablosuz sensör ağı akış şeması

Uygulamamızın amacı; hiyerarşik bir yapıda oluşturulan bir kablosuz sensör ağının ömrünü bulanık mantık kullanarak maksimize etmektir. Düğüm noktalarının sayısı ile küme sayısı kullanıcı tarafından değiştirilebilmektedir. Bu şekilde küme sayılarına göre de sistemin ömrünün karşılaştırılması da yapılabilmektedir.

5.2 Uygulamanın Adımları

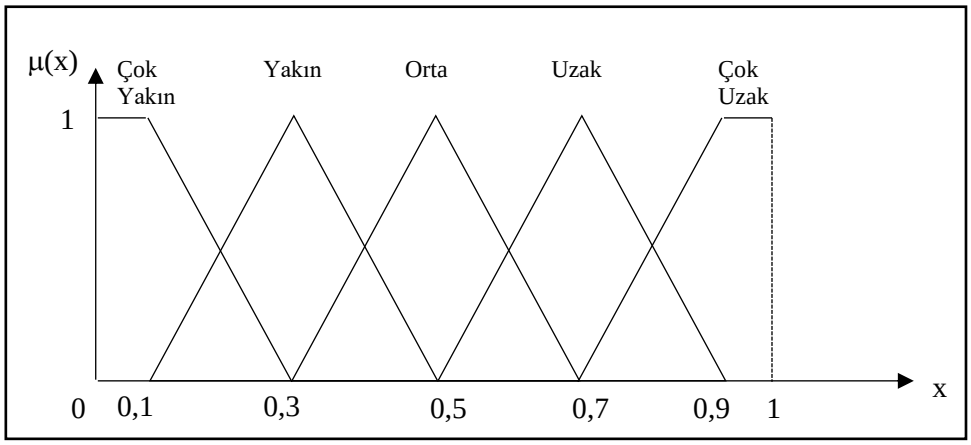
Adım 0: İlk olarak önceden belirtilen koordinat aralıklarında düğüm noktalarının koordinatlarını rastgele atanır. Her bir düğüm noktası için başlangıç enerji seviyesi (E_0) tespit edilir. Başlangıç enerji seviyesi tespit edilirken pillerdeki olası sapmalar da değerlendirilir. Uygulamamızda bu sapma %5 olarak değerlendirilmekte ve tüm düğüm noktalarının enerji seviyeleri bu sapma oranına göre rastgele atanmaktadır. K-ortalamalar algoritması kullanarak küme sayısı kadar düğüm noktası küme başı olarak hesaplanır ve adım 1'e geçilir.

Adım 1: Uygulamada Mamdani Bulanık Çıkarım Sistemi kullanılacaktır. Girdi değerleri olarak düğüm noktalarının enerji seviyeleri ve merkeze uzaklık değerleri ele alınacaktır. Bu değerler bulanık sayılar şeklinde sisteme girilecektir (Şekil 5.2 ve Şekil 5.3).

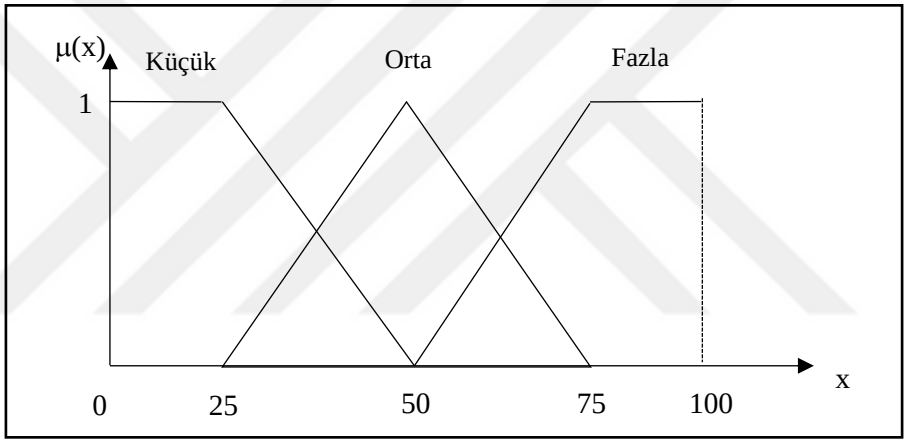


Şekil 5.2 Enerji seviyeleri bulanık değerleri

Girdi değerlerine göre kurallar oluşturulur ve düğüm noktalarının küme başı olabilme şansı hesaplanır. Küme başı olabilme şansları da bulanık sayılar şeklinde verilmektedir (Şekil 5.4).



Şekil 5.3 Merkeze yakınlık bulanık değerleri



Şekil 5.4 Düğüm noktalarının küme başı olma şansı için bulanık değerler

Düğüm noktalarının enerji seviyeleri (E) ve merkeze olan uzaklıklarına (C) göre küme başı olma şansını gösteren kurallar belirlenir:

- R_1 : E çok düşük ve C çok yakın ise Şans orta
- R_2 : E çok düşük ve C yakın ise Şans orta
- R_3 : E çok düşük ve C orta ise Şans küçük
- R_4 : E çok düşük ve C uzak ise Şans küçük
- R_5 : E çok düşük ve C çok uzak ise Şans küçük
- R_6 : E düşük ve C çok yakın ise Şans orta
- R_7 : E düşük ve C yakın ise Şans orta
- R_8 : E düşük ve C orta ise Şans küçük
- R_9 : E düşük ve C uzak ise Şans küçük

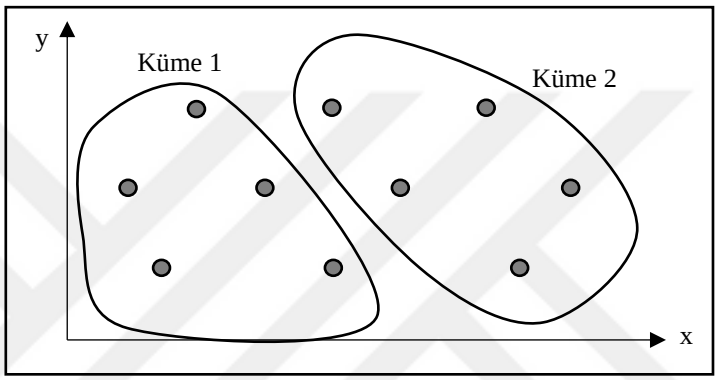
- R_{10} : E düşük ve C çok uzak ise Şans küçük
- R_{11} : E orta ve C çok yakın ise Şans fazla
- R_{12} : E orta ve C yakın ise Şans fazla
- R_{13} : E orta ve C orta ise Şans orta
- R_{14} : E orta ve C uzak ise Şans küçük
- R_{15} : E orta ve C çok uzak ise Şans küçük
- R_{16} : E yüksek ve C çok yakın ise Şans fazla
- R_{17} : E yüksek ve C yakın ise Şans fazla
- R_{18} : E yüksek ve C orta ise Şans fazla
- R_{19} : E yüksek ve C uzak ise orta
- R_{20} : E yüksek ve C çok uzak ise Şans orta
- R_{21} : E çok yüksek ve C çok yakın ise Şans fazla
- R_{22} : E çok yüksek ve C yakın ise Şans fazla
- R_{23} : E çok yüksek ve C orta ise fazla
- R_{24} : E çok yüksek ve C uzak ise Şans orta
- R_{25} : E çok yüksek ve C çok uzak ise Şans orta

Düğüm noktasının her bir kurala göre şansı hesaplanır ve sonra tüm kural çıktıları birleştirilerek sistemin yekûn bulanık çıktısı hesaplanır. Sonra WABL (weighted averaging based on levels) metodu kullanılarak bulanık çıktı kesin sayıya dönüştürülür. Nasiboglu ve Abdullayeva (2018)'de yaptıkları çalışmada yamuk bulanık sayıların WABL değerini hesaplamak için kullanışlı bir hesaplama yöntemi vermiştir. Bu çalışmada da WABL değerleri hesaplanırken bu yöntem kullanılmaktadır. Tüm düğüm noktaları için bu süreç tekrarlanır. Değerler hesaplandıktan sonra adım 2'ye geçilir.

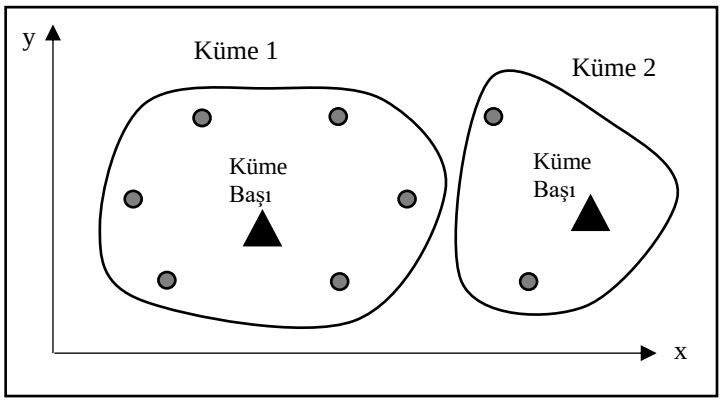
Adım 2: Her bir küme içinde şans değeri en yüksek olan düğüm noktası küme başı olarak seçilir. Düğüm noktalarının enerji sarfiyatını düşük tutmak maksadıyla, küme başı olan düğüm noktaları ile tüm düğüm noktaları arasındaki mesafe hesaplanır. Düğüm noktalarının küme başlarına olan mesafesine göre tekrar gruplandırma yapılır ve adım 3'e geçilir.

Örnek olarak; 10 düğüm noktası ve 2 kümenin olduğu bir sistem olsun. K-Ortalama Yöntemi ile elde edilen İlk gruplandırma aşağıdaki Şekil 5.5'de olduğu gibi olsun.

Her veri iletiminde enerji seviyeleri tekrar hesaplandığı için kümeler sürekli değişim göstermektedir. Her kümeleme sonrası Bulanık Çıkarım Sistemi ile küme başları tespit edilmektedir. Belli bir süre sonra yapılan gruplandırma Şekil 5.6'da olduğu gibidir.



Şekil 5.5 K-ortalama yöntemi ile yapılan gruplandırma çizelgesi



Şekil 5.6 Bulanık çıkarım sistemi ile yapılan gruplandırma çizelgesi

Adım 3: Bu adımda düğüm noktaları mesajlarını küme başlarına, küme başları da topladığı mesajları baz istasyonuna iletir. Daha önce de belirtildiği gibi enerji hesaplamaları küme başlarına mesaj ileten düğüm noktaları için kullanılan formül:

$$E_{Tx}(m,d) = \begin{cases} mE_{elec} + me_{fs}d^2, & d < d_0 \\ mE_{elec} + me_{mp}d^4, & d \geq d_0 \end{cases} \quad (5.1)$$

$$d_0 = \sqrt{\frac{e_{fs}}{e_{mp}}} \quad (5.2)$$

Baz istasyonlarına mesaj ileten küme başı düğüm noktaları için kullanılan formül:

$$E_{Tx}(m, d) = nm(E_{elec} + e_{fs}d_{bs} + E_{DA}) \quad (5.3)$$

Formülde kullanılan sabitler ve ve bu sabitlerin değeri ise aşağıdaki şekilde değerlendirilmiştir;

E_{elec} : 50 (nJ/bit)

e_{fs} : 10 (pJ/bit/m2)

e_{mp} : 0,0013 (pJ/bit/m4)

m : 4000 (bits)

d_{BS} : 1000 (m)

E_{DA} : 5 (nJ/bit/report)

Kalan enerji miktarı her bir düğüm noktası için hesaplandıktan sonra adım 4'e geçilir.

Adım 4: Kalan enerji seviyesi başlangıç enerji seviyesinin yüzde 5 (E_0*0,05) altında olan düğüm noktaları sistem dışına çıkartılır. Sistem içinde kalan düğüm noktalarının sayısı başlangıç düğüm noktası sayısının yüzde 10'un üzerindeyse kalan düğüm noktaları tekrar k-ortalamalar algoritması işlemine tabi tutularak işlemler tekrarlanır. Kalan düğüm noktası sayısı yüzde 10 ve altındaysa sistemin ömrü tamamlanmış olur ve işlem sonlandırılır.

5.3 Örnek Uygulama

5.3.1 Uygulamada Kullanılan Parametreler

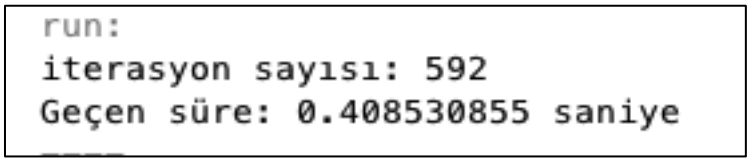
Örnek uygulamada kullanılan parametreler aşağıdaki gibidir. Bu parametreler kullanıcı tarafından değiştirilebilmektedir.

- Başlangıç aktif düğüm noktası sayısı: 100
- İstenilen küme başı düğüm noktası sayısı: 2

- Maksimum koordinat noktaları : (100,100)
- Minimum koordinat noktaları : (0,0)
- Düğüm noktalarının başlangıç enerji seviyesi (E_0) : 0,3 J (%5 yanılma payı)
- Düğüm noktalarının sistemde kalabilmesi için gerekli minimum enerji seviyesi: E_0 *0,05 J
- Sistemin çalışabilmesi için gerekli minimum düğüm noktası sayısı: Başlangıç aktif düğüm noktası sayısı*0,1
- Enerji dağılımı (E_{elec}) : 50 nJ/bit
- Verici amplifikatörü boş alan modeli (e_{fs}) : 10 pJ/bit/m2
- Verici amplifikatörü çok yollu modeli (e_{mp}) : 0,0013 pJ/bit/m4
- Veri paketi uzunluğu (M) : 4000 bits
- Merkez istasyona olan uzaklık (d_{BS}) : 1000 m
- Veri füzyonunda harcanan enerji (E_{DA}) : 5 nJ/bit/report

5.3.2 Uygulamanın Çalıştırılması

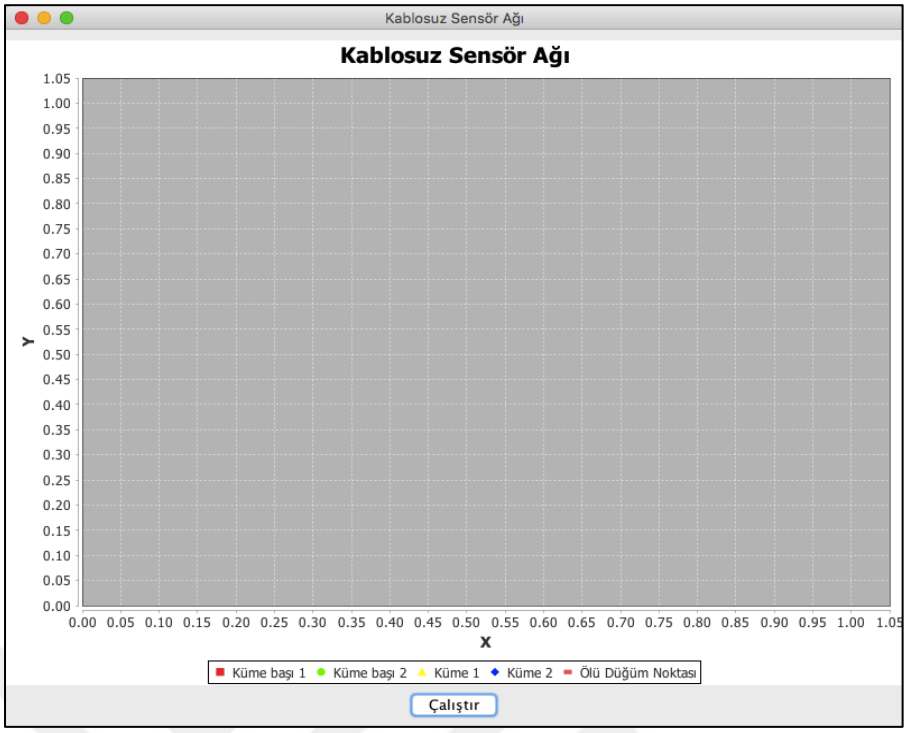
Örnek uygulama çalıştırıldığında ilk olarak bize sistemin çalışma süresini ve iterasyon sayısının vermektedir (Şekil 5.7).



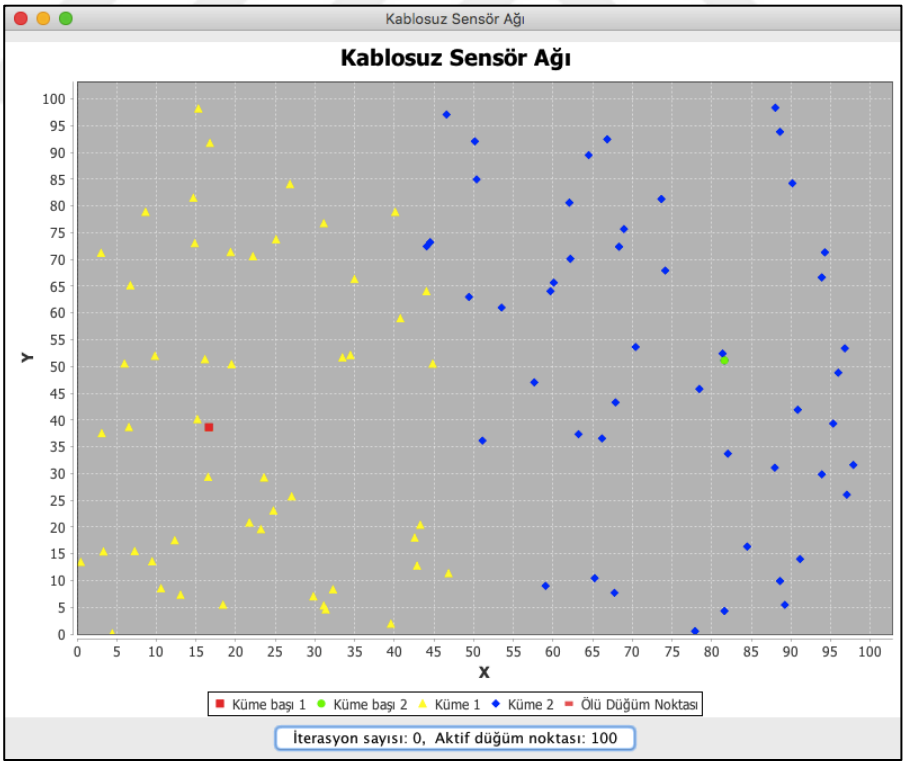
Şekil 5.7 Örnek uygulama değerleri

Uygulamadaki düğüm noktalarının görsel olarak gösterimi için yukarıdaki grafikteki "Çalıştır" butonuna basılır (Şekil 5.8).

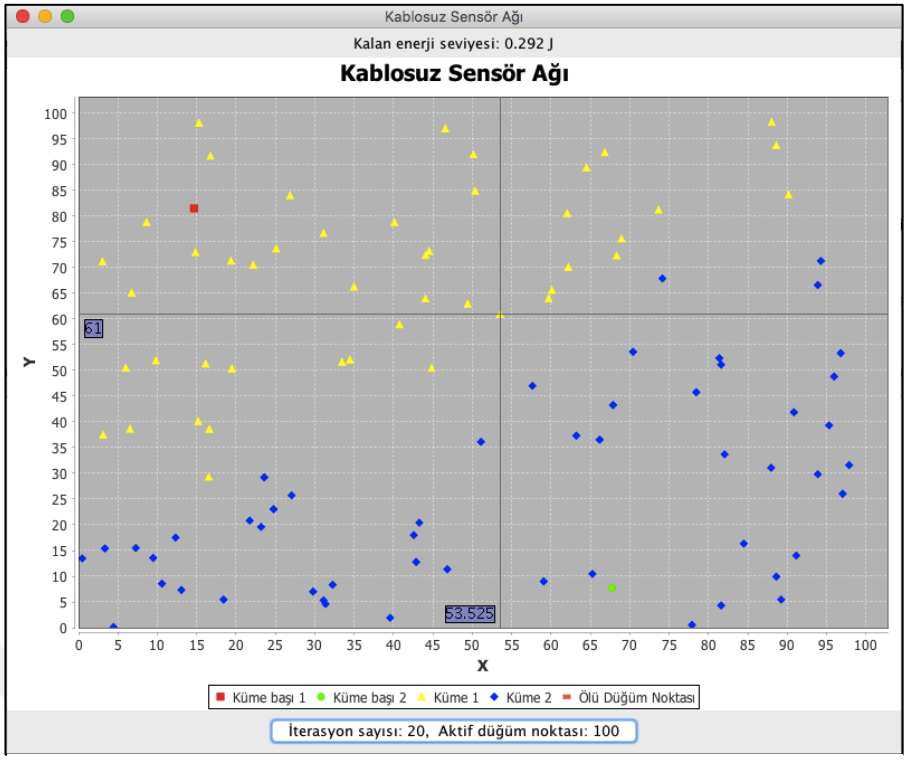
Grafikteki kırmızı kare birinci kümenin, yeşil daire ikinci kümenin küme başı düğüm noktalarını gösterir. Sarı üçgen birinci kümedeki düğüm noktalarını, mavi dörtgen ise ikinci kümedeki düğüm noktalarını temsil eder (Şekil 5.9 – 5.12).



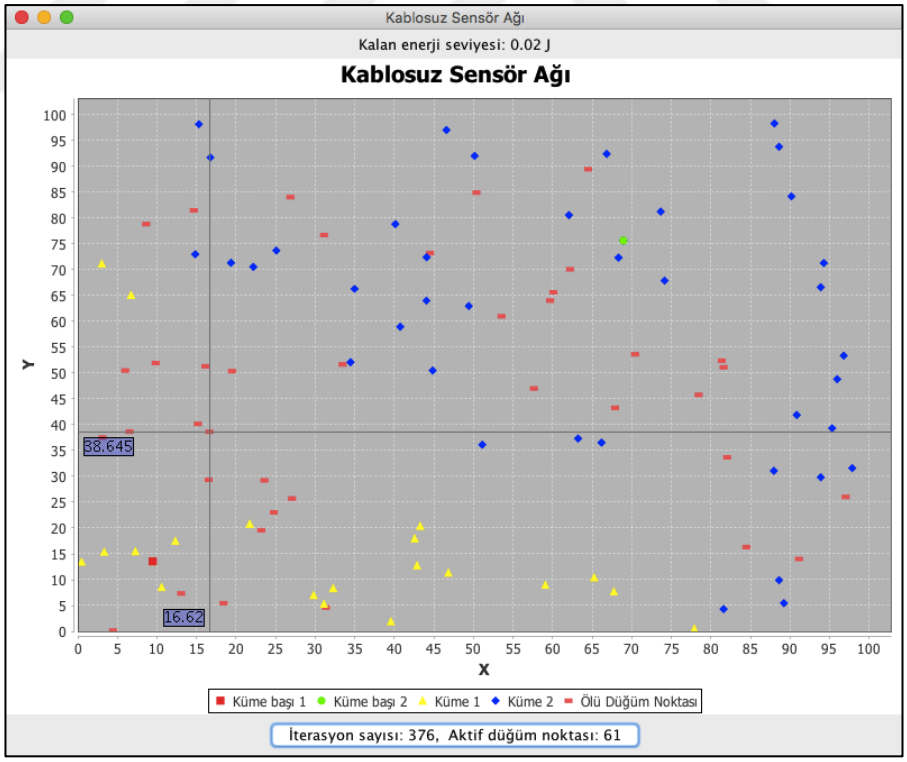
Şekil 5.8 Başlangıç grafiği



Şekil 5.9 Çalışan sistem grafiği 1



Şekil 5.10 Çalışan sistem grafiği 2

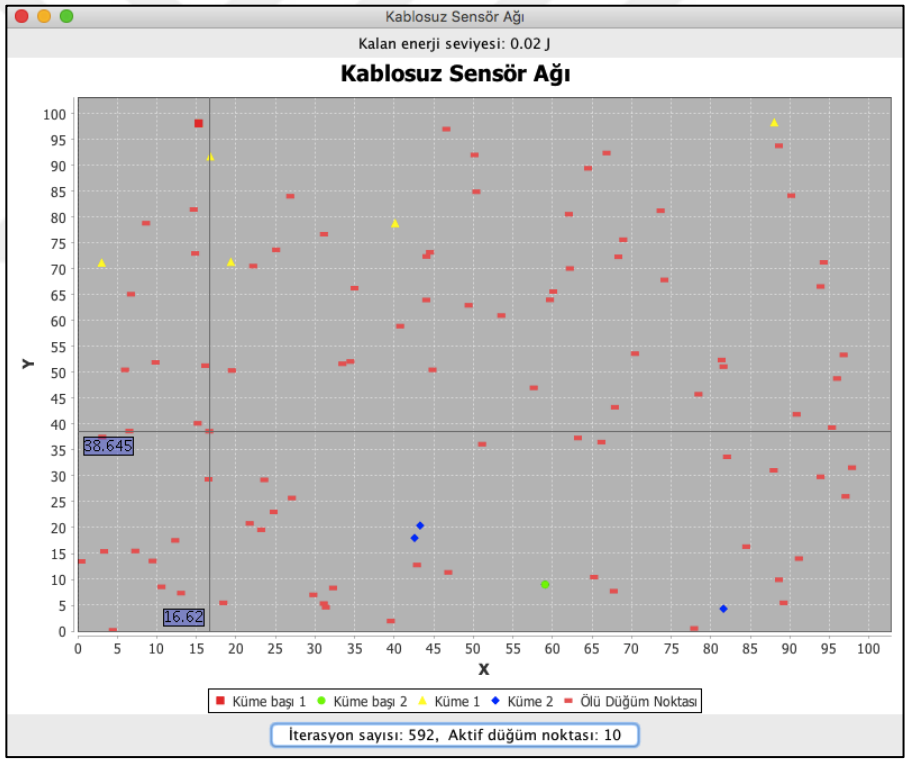


Şekil 5.11 Çalışan sistem grafiği 3

"Çalıştır" butonuna basıldığında iterasyon sayısı ve aktif düğüm noktaları sayısı eş zamanlı olarak gösterilmektedir. Butona basarak grafikteki hareketleri durdurabilir, tekrar basarak kaldığı yerden devam etmesi sağlanabilir.

Düğüm noktalarının üzerine çift tıklandığı zaman, koordinat değerleri ve o düğüm noktasına ait o andaki enerji seviyesi anlık olarak görülebilir.

Grafik incelendiğinde her iterasyonda kümelerin ve küme başı düğüm noktalarının değiştiği, enerji seviyesi yüzde 5'in altına düşen düğüm noktalarının da sistemden çıktığı görülmektedir. Ayrıca, sistemdeki aktif düğüm noktası sayısı yüzde 10'un altına düştüğünde sistemin çalışması durmaktadır. İterasyon sayısı bu kriter göre belirlenmektedir.



Şekil 5.12 Sistemin sonlandırıldığı halini gösteren grafik

Örnek uygulamaya ait değerler aşağıdaki tabloda sunulmuştur (Tablo 1). Tabloda, her iterasyon için aktif ve pasif düğüm noktası sayıları, minimum ve maksimum enerji seviyeleri ve küme başı düğüm noktalarının enerji seviyeleri görülmektedir.

Tablo 5.1 İterasyonlara göre kümelerdeki enerji değişimi

İterasyon Sayısı	Aktif Düğüm Nokta Sayısı		Pasif Düğüm Nokta Sayısı	Minimum Enerji Seviyesi		Maksimum Enerji Seviyesi		Küme Başı Düğüm Noktası Enerji Seviyesi	
	Küme 1	Küme 2		Küme 1	Küme 2	Küme 1	Küme 2	Küme 1	Küme 2
0	51	49	0	0,285	0,286	0,314	0,315	0,3	0,311
10	46	54	0	0,247	0,249	0,312	0,313	0,295	0,307
20	47	53	0	0,242	0,244	0,31	0,311	0,302	0,283
30	56	44	0	0,243	0,24	0,308	0,309	0,267	0,303
40	72	28	0	0,238	0,24	0,306	0,305	0,262	0,268
50	65	35	0	0,236	0,236	0,304	0,304	0,257	0,304
60	53	47	0	0,232	0,232	0,302	0,301	0,254	0,264
70	59	41	0	0,207	0,232	0,3	0,3	0,207	0,27
80	47	53	0	0,183	0,184	0,298	0,298	0,228	0,23
90	52	48	0	0,181	0,182	0,293	0,296	0,271	0,207
100	54	46	0	0,179	0,18	0,291	0,294	0,207	0,207
110	49	51	0	0,177	0,178	0,289	0,292	0,208	0,23
120	64	36	0	0,175	0,178	0,29	0,287	0,209	0,23
130	42	58	0	0,173	0,174	0,285	0,288	0,209	0,231
140	47	53	0	0,128	0,129	0,286	0,286	0,167	0,211
150	55	45	0	0,126	0,127	0,284	0,284	0,149	0,15
160	34	66	0	0,124	0,125	0,279	0,282	0,209	0,166
170	35	65	0	0,122	0,123	0,277	0,28	0,172	0,209
180	54	46	0	0,12	0,121	0,278	0,278	0,21	0,208
190	69	31	0	0,118	0,12	0,276	0,273	0,15	0,233
200	29	71	0	0,116	0,117	0,271	0,274	0,173	0,168
210	40	60	0	0,114	0,115	0,269	0,272	0,25	0,202
220	41	59	0	0,112	0,113	0,267	0,263	0,245	0,155
230	56	44	0	0,11	0,112	0,265	0,261	0,178	0,154
240	47	53	0	0,108	0,109	0,263	0,259	0,194	0,154
250	39	61	0	0,106	0,107	0,261	0,257	0,166	0,177
260	71	29	0	0,104	0,106	0,259	0,189	0,162	0,189
270	53	47	0	0,103	0,102	0,178	0,182	0,132	0,168
280	51	49	0	0,101	0,1	0,176	0,18	0,15	0,171
290	56	42	2	0,078	0,056	0,178	0,172	0,078	0,056
300	49	45	6	0,037	0,037	0,176	0,175	0,037	0,037
310	45	44	11	0,098	0,094	0,168	0,136	0,1	0,135
320	50	35	15	0,063	0,084	0,166	0,134	0,063	0,084

Tablo 5.1 devamı

330	27	53	20	0,09	0,066	0,136	0,164	0,09	0,082
340	41	35	24	0,078	0,064	0,162	0,13	0,099	0,069
350	28	44	28	0,092	0,062	0,132	0,16	0,103	0,09
360	20	49	31	0,067	0,06	0,13	0,158	0,09	0,06
370	19	46	35	0,065	0,046	0,128	0,156	0,065	0,046
380	30	30	40	0,053	0,056	0,154	0,122	0,053	0,069
390	22	34	44	0,058	0,042	0,152	0,12	0,058	0,042
400	32	21	47	0,074	0,06	0,126	0,118	0,088	0,06
410	37	13	50	0,055	0,084	0,124	0,116	0,055	0,084
420	15	32	53	0,07	0,076	0,113	0,122	0,099	0,099
430	25	20	55	0,068	0,057	0,12	0,112	0,077	0,057
440	29	13	58	0,043	0,072	0,118	0,11	0,043	0,072
450	25	15	60	0,037	0,045	0,116	0,108	0,037	0,045
460	18	19	63	0,039	0,061	0,105	0,114	0,039	0,078
470	22	12	66	0,057	0,042	0,112	0,103	0,067	0,042
480	26	5	69	0,058	0,032	0,11	0,102	0,09	0,032
490	13	15	72	0,056	0,07	0,108	0,1	0,082	0,086
500	18	9	73	0,044	0,068	0,106	0,098	0,065	0,084
510	19	6	75	0,042	0,076	0,104	0,096	0,067	0,082
520	18	5	77	0,04	0,064	0,102	0,094	0,073	0,074
530	16	5	79	0,038	0,062	0,1	0,092	0,084	0,077
540	13	8	79	0,036	0,05	0,098	0,09	0,038	0,05
550	15	3	82	0,044	0,068	0,096	0,069	0,044	0,069
560	10	7	83	0,042	0,031	0,085	0,094	0,047	0,031
570	7	8	85	0,064	0,035	0,092	0,083	0,068	0.035
580	7	5	88	0,039	0,037	0,09	0,081	0,039	0,037
590	6	4	90	0,06	0,035	0,076	0,079	0,076	0,035
592	6	4	90	0,06	0,031	0,072	0,078	0,072	0,031

BÖLÜM ALTI
SONUÇLAR

Kablosuz sensör ağları günümüzde birçok alanda kullanılmaktadır ve düğüm noktalarının enerji kullanımın en aza indirgenmesi sistemin ömrü için büyük bir önem taşımaktadır. Sistem ömrünün uzatılması için çeşitli yöntemler denenmiş ve bu yöntemlerden bazıları günümüz kablosuz sensör ağlarında hali hazırda kullanılmaktadır.

Yapılan bu araştırma ve uygulamada Seviyelere Dayalı Ağırlıklı Ortalama Temelli Bulanık Çıkarsama Yöntemi kullanılarak bir kablosuz sensör ağının çalışması incelenmektedir. Düğüm noktalarının merkeze yakınlık ve enerji seviyeleri dilsel değişkenlere çevrilmektedir. Mamdani Bulanık Çıkarım Sistemi kullanılarak her bir düğüm noktasının küme başı olma şans değeri elde edilmektedir. Bu değerin en yüksek olduğu düğüm noktaları küme başı olarak seçilmekte ve kümeler oluşturulmaktadır.

Her bir veri iletiminde düğüm noktalarının küme başı olma şans değerleri tekrar hesaplanmakta, küme başı düğüm noktaları tekrar belirlenmekte ve K-ortalamalar algoritması ile kümeler tekrar oluşturulmaktadır. Bu dinamik yapı sayesinde düğüm noktalarının enerji seviyelerindeki fark en aza indirgenmekte ve her bir düğüm noktasının sistemde kalma süresini büyük ölçüde arttırmaktadır.

Uygulamada düğüm noktası sayısı ve küme sayısı önceden belirlenmektedir. Düğüm noktalarının belirlenen diğer parametreleri girildikten sonra program çalıştırılmaktadır. Düğüm noktalarının enerji seviyeleri her bir iterasyonda hesaplanmakta, enerji seviyesi başlangıç enerji seviyesinin yüzde 5'inden az olan düğüm noktaları sistem dışı bırakılmaktadır. Sistem içi olan düğüm noktalarının sayısı toplam düğüm noktası sayısının yüzde 10'undan az ise sistem ömrünü tamamlamakta ve kapatılmaktadır.

Çalıştırılan uygulama ile kablosuz sensör ağlarının kullanım ömrünün belirlenmesinde Seviyelere Dayalı Ağırlıklı Ortalama Temelli Bulanık Çıkarsama Yönteminin etkili bir şekilde kullanılabileceği sonucuna varılmaktadır. Küme başı düğüm noktalarının seçiminde dilsel değişkenlerden faydalanılmaktadır. Karar verici, dilsel değişken olan girdi değerleri ile yine bir dilsel değişken olan çıktı değerini belirleyebilmektedir. Bu durum karar verici için çok büyük bir kolaylık sağlamaktadır. Matematiksel olarak hesaplama yapmadan daha anlaşılır bilgiler üzerinden karar verici kararını verebilmektedir.

Sonuç olarak; bulanık mantık ve bulanık çıkarım sistemleri birçok alanda olduğu gibi kablosuz sensör ağlarının çalıştırılmasında ve kullanım ömrünün belirlenmesinde de etkin olarak kullanılabilmektedir. Çalıştırılan uygulama ile;

- Kablosuz sensör ağının başarılı bir şekilde çalıştığı, düğüm noktalarının enerji kullanım dengesinin etkin bir şekilde yapıldığı görülmektedir.
- Programın görsel grafiği incelendiğinde küme başı düğüm noktalarının merkeze yakın bölgelerden seçildiği görülmektedir.
- Küme başı olan düğüm noktaları arasındaki mesafenin mantıklı bir uzaklıkta ve kararlı olduğu görülmektedir.
- Kümelerin küme başı düğüm noktalarının etrafında oluştuğu görülmektedir.
- Grafik üzerinde, düğüm noktalarının kalan enerji seviyeleri anlık olarak gözlenebilmekte, diğer düğüm noktalarıyla paralel oranda azalış olduğu görülmektedir.
- Enerji seviyesi istenilen seviyenin altına düşen düğüm noktalarının sistem dışı kaldığı görülmektedir.
- Sistem içinde kalan düğüm noktalarının sayısı belli bir seviyenin altına indiği zaman sistemin durduğu görülmektedir.

KAYNAKLAR

Balakrishnan, B. ve Balachandran, S. (2017). FLECH: fuzzy logic based energy efficient clustering hierarchy for nonuniform wireless sensor networks. *Wireless Communications and Mobile Computing*, *2017*.

Bidaki, M. ve Tabbakh, R. K. (2016). Efficient Fuzzy Logic-Based Clustering Algorithm for Wireless Sensor Networks. *International Journal of Grid and Distributed Computing*, *9*(5), 79-88.

El Alami, H. ve Najid, A. (2017). Fuzzy Logic Based Clustering Algorithm for Wireless Sensor Networks. *International Journal of Fuzzy System Applications (IJFSA)*, *6*(4), 63-82.

El Alami, H. ve Najid, A. (2015). SEFP: A new routing approach using fuzzy logic for clustered heterogeneous wireless sensor networks. *International Journal on Smart Sensing & Intelligent Systems*, *8*(4), 2286-2306.

Gupta, I., Riordan, D. ve Sampalli, S. (2005). Cluster-head election using fuzzy logic for wireless sensor networks. *Communication Networks and Services Research Conference. Proceedings of the 3rd Annual*, 255-260.

Jain, N., Gupta, S. K. ve Sinha, P. (2013). Clustering protocols in wireless sensor networks: A survey. *International Journal of Applied Information System (IJAIS)*, *5*(2), 41-50.

Jang, J. S. R., Sun, C. T. ve Mizutani, E. (1997). *Neuro-fuzzy and soft computing; a computational approach to learning and machine intelligence*. Upper Saddle River: Prentice-Hall.

Iancu, I. (2012). A Mamdani type fuzzy logic controller. *Fuzzy Logic-Controls, Concepts, Theories and Applications*. IntechOpen.

Larose, Daniel T. (2005). *Discovering Knowledge in Data-An Introduction to Data Mining*. New Jersey: John Wiley & Sons.

Işık, M. ve Çamurcu, A. Y. (2010). K-Means ve Aşırı Küresel C-Means Algoritmaları ile Belge Madenciliği. *Marmara Fen Bilimleri Dergisi, 22*(1), 1-18.

Mehra, P. S., Doja, M. N. ve Alam, B. (2018). Fuzzy based enhanced cluster head selection (FBECS) for WSN. *Journal of King Saud University-Science*, 25 April 2018, 1-12.

Mert, A. (2006). *Bulanık Bilgilerde Seviyelere Dayalı Ağırlıklı Ortalama Yöntemlerine İlişkin Araştırmalar ve Uygulamaları*. Doktora Tezi, Ege Üniversitesi Fen Bilimleri Enstitüsü, İzmir.

Muhammed Amin Murad A. (2016). *Kablosuz Algılayıcı Ağlarda Kümeleme Algoritmaları ile Enerji Verimliliğinin Arttırılması İçin Alternatif Bir Yöntem Geliştirme.* Doktora Tezi, Gazi Üniversitesi Bilişim Enstitüsü, Ankara.

Najmeh Kamyab Pour (2015). *Energy Efficiency in Wireless Sensor Networks*. Phd Thesis, University of Technology, Sydney.

Nasiboglu R. ve Abdullayeva R. (2018). Analytical Formulations for the Level Based Weighted Average Value of Discrete Trapezoidal Fuzzy Numbers. *International Journal on Soft Computing (IJSC)*, 2018, (9) No.2/3, 1-15.

Nasibov, E. N. ve Mert, A. (2007). On methods of defuzzification of parametrically represented fuzzy numbers. *Automatic Control and Computer Sciences*, *41*(5), 265-273.

Nehra, V., Pal, R. ve Sharma, A. K. (2013). Fuzzy-based leader selection for topology controlled PEGASIS protocol for lifetime enhancement in wireless sensor network. *International Journal of Computers & Technology*, *4*(3), 755-764.

Nithya, S. ve Gomathy, C. (2015). A Survey of Attacks in Wireless Sensor Network. *International Journal of Applied Engineering Research*, *10*(26), 21531-21538.

Priyadarshi, R., Lucky Singh, R., Sharma, I. ve Kumar, S. (2018). Energy Efficient LEACH Routing in Wireless Sensor Network. *International Journal of Pure and Applied Mathematics*, *118*(20), 2018: 135-142.

Sert, S. A., Yazici, A. ve Dokeroglu, T. (2016). Fuzzy processing in surveillance wireless sensor networks. *IEEE International Conference on Fuzzy Systems (FUZZ-IEEE)*, 1509-1515.

Sharma, T. ve Kumar, B. (2012). F-MCHEL: Fuzzy based master cluster head election leach protocol in wireless sensor network. *International Journal of Computer Science and Telecommunications*, *3*(10), 8-13.

Sharma, D., Verma, S. ve Sharma, K. (2013). Network topologies in wireless sensor networks: a review. *International Journal of Electronics and Communication Technology*, *4*(3), 93-97.

Wang, J., Yang, X., Zheng, Y., Zhang, J. ve Kim, J. U. (2012). An energy-efficient multi-hop hierarchical routing protocol for wireless sensor networks. *International Journal of Future Generation Communication and Networking*, *5*(4), 89-98.

Zhang, Y., Wang, J., Han, D., Wu, H. ve Zhou, R. (2017). Fuzzy-logic based distributed energy-efficient clustering algorithm for wireless sensor networks. *Sensors*, *17*(7), 1554.

EKLER

EK 1: Uygulamada Kullanılan Programın Kodları

```
package wireless_sensor;
import java.awt.BasicStroke;
import java.awt.BorderLayout;
import java.awt.Color;
import java.awt.Dimension;
import java.awt.EventQueue;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.awt.geom.Rectangle2D;
import static java.lang.Double.min;
import java.text.DecimalFormat;
import java.util.*;
import javax.swing.JButton;
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JPanel;
import org.apache.commons.lang3.ArrayUtils;
import org.jfree.chart.axis.ValueAxis;
import org.jfree.chart.plot.PlotOrientation;
import org.jfree.chart.plot.XYPlot;
import org.jfree.chart.renderer.xy.XYItemRenderer;
import org.jfree.data.xy.XYDataItem;
import org.jfree.data.xy.XYDataset;
import org.jfree.data.xy.XYSeries;
import org.jfree.data.xy.XYSeriesCollection;
import org.jfree.ui.RectangleEdge;
import org.jfree.chart.ChartPanel;
import org.jfree.chart.panel.CrosshairOverlay;
import org.jfree.chart.plot.Crosshair;
```

```
import org.jfree.chart.ChartMouseEvent;
import org.jfree.chart.ChartMouseListener;
import org.jfree.chart.ChartFactory;
import org.jfree.chart.JFreeChart;
/**
 * @author tekinerten
 */
public class Wireless_sensor extends JFrame implements ChartMouseListener{
    private static int starting_nodes = 100; // count of nodes
    int k = 2 ; // cluster head count for k-means algorithm
    private static double E0 = 0.3; // Initial energy (J)
    private static double min_energy_level = E0*0.1;
    private static final int min_node_level = (int) (starting_nodes * 0.1);
    /*  max and min coordinates   */
    double max_x = 100;
    double max_y = 100;
    double min_x = 0;
    double min_y = 0;
    /*  WABL parameters  */
    double c = 0.5; // optimism coefficient
    /*------------------------------------------------------------ */
    private static final String title = "Kablosuz Sensör Ağı";
    private XYSeries moved = new XYSeries("Düğüm Noktaları");
    private XYSeries cluster_heads = new XYSeries("Baş Düğüm Noktaları");
    private XYSeries[] series;
    private XYSeries[] ch_series;
    private XYSeries dead_series;
    /*------------------------------------------------------------ */
    static int E_elec = 50; //  Energy dissipation to run the radio device (nJ/bit)
    static int e_fs = 10; // Free space model of transmitter amplifier (pJ/bit/m2)
    static double e_mp = 0.0013  ;  // Multi-path model of transmitter amplifier
(pJ/bit/m4 )
```

```
    static double d0 = Math.sqrt(e_fs/e_mp); // Distance threshold  (m)
    static int E_DA = 5; // Energy exhausted in data fusion (nJ/bit/report)
    static int M = 4000;  // Data packet size(bits)
    static int d_BS = 1000;  //  distance to base station (m)
    /*---------------------------------------------------------- */
    int iteration = 0;
    int nodes = starting_nodes;
    double[][] m, coord, distance_to_mean, distance_to_ch;
    int[]  cluster_head, ch_array,membership_array,mean_membership,
cluster_membership;
    double SSE;
    double[] min_distance_to_mean,residual_energy, min_distance_to_ch,
          crisp_value,nodes_energy_array ;
    double[] energy_v_low, energy_low, energy_medium, energy_high,
energy_v_high;
    double[] centr_v_close, centr_close, centr_medium, centr_far, centr_v_far;
    double v_low_en, low_en, medium_en, high_en, v_high_en;
    double v_close_cent,close_cent,medium_cent,far_cent, v_far_cent;
    double[][] chance, max_chance;
    double[][][] chance_value;
    long startTime,endTime;
    DecimalFormat df = new DecimalFormat("#.###");
    /*---------------------------------------------------------- */
    JButton button;
    JLabel res_energy;
    private int counter = 0;
    private javax.swing.Timer timer;
    /*---------------------------------------------------------- */
    ArrayList <Integer> ch_list;
    ArrayList<double[]> dead_coord = new ArrayList<>();
    ArrayList<Double> dead_energy = new ArrayList<>();
    TreeMap<Integer, int[]> head_list =  new TreeMap<>();
```

```
    TreeMap<Integer, double[][]> coord_list =  new TreeMap<>();
    TreeMap<Integer, int[]> ch_membership_list =  new TreeMap<>();
    TreeMap<Integer, double[][]> dead_coord_list =  new TreeMap<>();
    TreeMap<Integer, double[]> energy_list =  new TreeMap<>();
    TreeMap<Integer, Double[]> dead_energy_list =  new TreeMap<>();
    TreeMap<Integer, Double> ch_energy;
    TreeMap<Integer, TreeMap<Integer,Double>> ch_energy_list =  new
TreeMap<>();
    TreeMap<Integer, ArrayList<Double>> cluster_energy;
    TreeMap<Integer, TreeMap<Integer,ArrayList<Double>>> cluster_energy_list =
new TreeMap<>();
    /*------------------------------------------------------------ */
    private ChartPanel chartPanel;
    private Crosshair xCrosshair;
    private Crosshair yCrosshair;
    Rectangle2D dataArea;
    XYPlot xyPlot;

    public Wireless_sensor(String s) {
        super(s);
        set_chart();
        startTime = System.nanoTime();
        adim_0();
    }
    /**
     * @param args the command line arguments
     */
    public static void main(String[] args) {
        EventQueue.invokeLater(new Runnable() {
            @Override
            public void run() {
                Wireless_sensor chart = new Wireless_sensor(title);
```

```
            chart.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
            chart.pack();
            chart.setLocationRelativeTo(null);
            chart.setVisible(true);
        }
    });
}
private void adim_0() {
    coord = new double[nodes+1][3];
    for(int i=1; i<=nodes; i++){
        coord[i][1] = (float) (Math.random() * (max_x - min_x) + min_x);
        coord[i][2] = (float) (Math.random() * (max_y - min_y) + min_y);
    }
    residual_energy = new double[nodes+1];
    Random en_rate = new Random();
    for(int i=1; i<=nodes; i++){
        double num = en_rate.nextDouble()*(1.05-0.95) + 0.95;
        residual_energy[i] = E0*num;
    }
    m = new double[k+1][3];
    for(int i=1; i<=k; i++){
        m[i][1] = (Math.random() * (max_x - min_x) + min_x);
        m[i][2] = (Math.random() * (max_y - min_y) + min_y);
    }
    k_means_1();
}
private void adim_1(){
    energy_v_low = new double[nodes+1];
    energy_low = new double[nodes+1];
    energy_medium = new double[nodes+1];
    energy_high = new double[nodes+1];
    energy_v_high = new double[nodes+1];
```

```
    centr_v_close = new double[nodes+1];
    centr_close = new double[nodes+1];
    centr_medium = new double[nodes+1];
    centr_far = new double[nodes+1];
    centr_v_far = new double[nodes+1];
    // calculate u values for all nodes
    for(int i= 1; i<=nodes; i++){
        get_linguistic_value(residual_energy[i],min_distance_to_mean[i]);
        energy_v_low[i] = v_low_en;
        energy_low[i] = low_en;
        energy_medium[i] = medium_en;
        energy_high[i] = high_en;
        energy_v_high[i] = v_high_en;
        centr_v_close[i] = v_close_cent;
        centr_close[i] = close_cent;
        centr_medium[i] = medium_cent;
        centr_far[i] = far_cent;
        centr_v_far[i] = v_far_cent;
    }
    fuzzy_rule_base();
    adim_2();
}
private void adim_2(){
   cluster_head = new int[k+1];
   ch_list = new ArrayList<>();
   for(int i=1;i<=k;i++){
       double max = 0;
       int ch = 0;
       for(int j=1; j<=nodes; j++){
           if(mean_membership[j] == i && crisp_value[j]>max){
               max = crisp_value[j];
               ch = j;
```

```
            }
        }
        ch_list.add(ch);
        cluster_head[i] = ch;
    }
    set_cluster_membership();
    adim_3();
}
private void adim_3(){
    ch_energy = new TreeMap <>();
    for(int i=1; i<=nodes; i++){
        if(ch_list.contains(i)){
            for(int j=1; j<=k; j++){
                if(cluster_head[j] == i){
                    ch_energy.put(j,residual_energy[i]);
                }
            }
        }
    }
    ch_energy_list.put(iteration, ch_energy);
   /*------------------------------------------------------------ */
   cluster_energy = new TreeMap <>();
    for(int i=1; i<=k; i++){
        ArrayList<Double> cluster_en = new ArrayList<>();
        for(int j=1; j<=nodes; j++){
            if(cluster_membership[j]==i){
                cluster_en.add(residual_energy[j]);
            }
        }
        cluster_energy.put(i,cluster_en);
    }
    cluster_energy_list.put(iteration, cluster_energy);
```

```
        /*------------------------------------------------------------ */
        double[] res_energy_array = ArrayUtils.remove(residual_energy,0);
        energy_list.put(iteration, res_energy_array);
        for(int i=1; i<=nodes; i++){
            if(ch_list.contains(i)){
                residual_energy[i] = residual_energy[i] -
ch_energy_level(min_distance_to_ch[i]);
            }
            else{
                residual_energy[i] = residual_energy[i] -
node_energy_level(min_distance_to_ch[i]);
            }
        }
        adim_4();
    }
    private void adim_4(){
        ch_array = ArrayUtils.removeAllOccurences(cluster_head,0);
        head_list.put(iteration, ch_array);
        /*------------------------------------------------------------ */
        double[][] coord_array = ArrayUtils.remove(coord,0);
        coord_list.put(iteration, coord_array);
        /*------------------------------------------------------------ */
        int[] ch_memb_array = ArrayUtils.remove(cluster_membership,0);
        ch_membership_list.put(iteration, ch_memb_array);
        /*------------------------------------------------------------ */
        double[][] dead_coord_arr = new double[dead_coord.size()][3];
        dead_coord_arr = dead_coord.toArray(dead_coord_arr);
        dead_coord_list.put(iteration, dead_coord_arr);
        /*------------------------------------------------------------ */
        Double[] dead_energy_arr = new Double[dead_energy.size()];
        dead_energy_arr = dead_energy.toArray(dead_energy_arr);
        dead_energy_list.put(iteration, dead_energy_arr);
```

```
        /*------------------------------------------------------------ */
        for(int i=1; i<=nodes; i++){
           if(residual_energy[i] < min_energy_level){
              dead_coord.add(coord[i]);
              dead_energy.add(residual_energy[i]);
              coord = ArrayUtils.remove(coord,i);
              residual_energy = ArrayUtils.remove(residual_energy,i);
              nodes --;
           }
        }
        if(nodes>=min_node_level){
           iteration ++;
           k_means_1();
        }
        else{
           endTime   = System.nanoTime();
           double totalTime = ((double)(endTime - startTime)/ 1000000000); //
nanoseconds to second
           System.out.println("iterasyon sayısı: "+iteration);
           System.out.println("Geçen süre: "+totalTime+" saniye");
           System.out.println("----");
        }
     }
     private void set_cluster_membership(){
        cluster_membership = new int[nodes+1];
        distance_to_ch = new double[nodes+1][k+1];
        min_distance_to_ch = new double[nodes+1];
        /*------------------------------------------------------------ */
        for(int i=1; i<=nodes; i++){
           for(int j=1; j<=k; j++){
              distance_to_ch[i][j] = euclidean_distance(coord[i][1],
coord[cluster_head[j]][1],coord[i][2],coord[cluster_head[j]][2]);
```

```
        }
        if(!ch_list.contains(i)){
            distance_to_ch[i] = ArrayUtils.removeAllOccurences(distance_to_ch[i],0);
            distance_to_ch[i] = ArrayUtils.insert(0,distance_to_ch[i],0);
        }
    }
    for(int i=1; i<=nodes; i++){
        min_distance_to_ch[i] =  getMin(distance_to_ch[i]);
        for(int j=1; j<=k; j++){
            if(min_distance_to_ch[i] == distance_to_ch[i][j]){
                cluster_membership[i] = j;
            }
        }
    }
}
private void get_linguistic_value(double en,double cent){
    // energy
    if(en <= E0*0.1){
        v_low_en = 1;
    }
    else if(en > E0*0.1 && en <= E0*0.3){
        v_low_en = (E0*0.3-en)/(E0*0.2);
    }
    else{
        v_low_en = 0;
    }
    if(en > E0*0.1 && en <= E0*0.3){
        low_en = (en-E0*0.1)/(E0*0.2);
    }
    else if(en > E0*0.3 && en <= E0*0.5){
        low_en = (E0*0.5-en)/(E0*0.2);
    }
```

```
else{
   low_en = 0;
}
if(en > E0*0.3 && en <= E0*0.5){
   medium_en = (en-E0*0.3)/(E0*0.2);
}
else if(en > E0*0.5 && en <= E0*0.7){
   medium_en = (E0*0.7-en)/(E0*0.2);
}
else{
   medium_en = 0;
}
if(en >E0*0.5 && en <= E0*0.7){
   high_en = (en-E0*0.5)/(E0*0.2);
}
else if(en >E0*0.7 && en <= E0*0.9){
   high_en = (E0*0.9-en)/(E0*0.2);
}
else{
   high_en = 0;
}
 if(en >E0*0.7 && en <= E0*0.9){
   v_high_en = (en-E0*0.7)/(E0*0.2);
}
else if(en > E0*0.9){
   v_high_en = 1;
}
else{
   v_high_en = 0;
}
// centrality
```

```
double max_dist = Math.sqrt(Math.pow(max_x-min_x,2)+Math.pow(max_y-
min_y,2));
if(cent <= max_dist*0.1){
   v_close_cent = 1;
}
else if(cent > max_dist*0.1 && cent <= max_dist*0.3){
   v_close_cent = (max_dist*0.3-cent)/(max_dist*0.2);
}
else{
   v_close_cent = 0;
}
if(cent > max_dist*0.1 && cent <= max_dist*0.3){
   close_cent = (cent-max_dist*0.1)/(max_dist*0.2);
}
else if(cent > max_dist*0.3 && cent <= max_dist*0.5){
   close_cent = (max_dist*0.5-cent)/(max_dist*0.2);
}
else{
   close_cent = 0;
}
if(cent > max_dist*0.3 && cent <= max_dist*0.5){
   medium_cent = (cent-max_dist*0.3)/(max_dist*0.2);
}
else if(cent > max_dist*0.5 && cent <= max_dist*0.7){
   medium_cent = (max_dist*0.7-cent)/(max_dist*0.2);
}
else{
   medium_cent = 0;
}
if(cent >max_dist*0.5 && cent <= max_dist*0.7){
   far_cent = (cent-max_dist*0.5)/(max_dist*0.2);
}
```

```
    else if(cent >max_dist*0.7 && cent <= max_dist*0.9){
        far_cent = (max_dist*0.9-cent)/(max_dist*0.2);
    }
    else{
        far_cent = 0;
    }
     if(cent >max_dist*0.7 && cent <= max_dist*0.9){
        v_far_cent = (cent-max_dist*0.7)/(max_dist*0.2);
    }
    else if(cent >max_dist*0.9){
        v_far_cent = 1;
    }
    else{
        v_far_cent = 0;
    }
}
private void fuzzy_rule_base(){
    chance = new double[nodes+1][28];
    for(int i=1; i<=nodes; i++){
        chance[i][1] =  min(energy_v_low[i],centr_v_close[i]);
        chance[i][2] =  min(energy_v_low[i],centr_close[i]);
        chance[i][3] =  min(energy_v_low[i],centr_medium[i]);
        chance[i][4] =  min(energy_v_low[i],centr_far[i]);
        chance[i][5] =  min(energy_v_low[i],centr_v_far[i]);
        chance[i][6] =  min(energy_low[i],centr_v_close[i]);
        chance[i][7] =  min(energy_low[i],centr_close[i]);
        chance[i][8] =  min(energy_low[i],centr_medium[i]);
        chance[i][9] =  min(energy_low[i],centr_far[i]);
        chance[i][10] =  min(energy_low[i],centr_v_far[i]);
        chance[i][11] =  min(energy_medium[i],centr_v_close[i]);
        chance[i][12] =  min(energy_medium[i],centr_close[i]);
        chance[i][13] =  min(energy_medium[i],centr_medium[i]);
```

```
            chance[i][14] =  min(energy_medium[i],centr_far[i]);
            chance[i][15] =  min(energy_medium[i],centr_v_far[i]);
            chance[i][16] =  min(energy_high[i],centr_v_close[i]);
            chance[i][17] =  min(energy_high[i],centr_close[i]);
            chance[i][18] =  min(energy_high[i],centr_medium[i]);
            chance[i][19] =  min(energy_high[i],centr_far[i]);
            chance[i][20] =  min(energy_high[i],centr_v_far[i]);
            chance[i][21] =  min(energy_v_high[i],centr_v_close[i]);
            chance[i][22] =  min(energy_v_high[i],centr_close[i]);
            chance[i][23] =  min(energy_v_high[i],centr_medium[i]);
            chance[i][24] =  min(energy_v_high[i],centr_far[i]);
            chance[i][25] =  min(energy_v_high[i],centr_v_far[i]);
        }
        get_wabl();
    }
    private void get_wabl(){
        crisp_value = new double[nodes+1];
        double u_kucuk,u_orta,u_fazla = 0;
        double WABL_1,WABL_2,WABL_3,WABL = 0;
        /*------------------------------------------------------------ */
        for(int i=1; i<=nodes; i++){
            u_kucuk = Math.max(Math.max(Math.max(chance[i][3],
Math.max(chance[i][4],chance[i][5])),
                               Math.max(chance[i][8],
Math.max(chance[i][9],chance[i][10]))),
                                 Math.max(chance[i][14],chance[i][15]));

            u_orta = Math.max(Math.max(Math.max(chance[i][1],
Math.max(chance[i][2],chance[i][6])),
                               Math.max(chance[i][7],
Math.max(chance[i][13],chance[i][19]))),
```

```
Math.max(chance[i][20],Math.max(chance[i][24],chance[i][25])));

        u_fazla = Math.max(Math.max(Math.max(chance[i][11],
Math.max(chance[i][12],chance[i][16])),
                           Math.max(chance[i][17],
Math.max(chance[i][18],chance[i][21]))),
                             Math.max(chance[i][22],chance[i][23]));
        /*------------------------------------------------------------ */
        if(u_fazla>0 && u_orta==0 && u_kucuk==0 ){
           WABL = ((1-c)*(50+50+u_fazla*25)+c*(100+100))/2;
        }
        else if(u_fazla==0 && u_orta>0 && u_kucuk==0 ){
           WABL = ((1-c)*(25+25+u_orta*25)+c*(75-u_orta*25+75))/2;
        }
        else if(u_fazla==0 && u_orta==0 && u_kucuk>0 ){
           WABL = ((1-c)*0 + c*(50-u_kucuk*25+50))/2;
        }
        else if(u_fazla>0 && u_orta>0 && u_kucuk==0 ){
           if(u_fazla>u_orta){
              WABL_1 = ((1-c)*(25+25+u_orta*25)+c*(100+100))/2;
              WABL_2 = ((1-c)*(50+u_orta*25+50+u_fazla*25)+c*(100+100))/2;
              WABL = u_orta*WABL_1 + (u_fazla-u_orta)*WABL_2;
           }
           else{
              WABL_1 = ((1-c)*(25+25+u_fazla*25)+c*(100+100))/2;
              WABL_2 = ((1-c)*(25+u_fazla*25+25+u_orta*25)+c*(75-
u_orta*25+75-u_fazla*25))/2;
              WABL = u_fazla*WABL_1 + (u_orta-u_fazla)*WABL_2;
           }
        }
        else if(u_fazla==0 && u_orta>0 && u_kucuk>0 ){
```

```
            if(u_orta>u_kucuk){
               WABL_1 = ((1-c)*(0)+c*(75-u_kucuk*25+75))/2;
               WABL_2 = ((1-c)*(25+u_kucuk*25+25+u_orta*25)+c*(75-
u_orta*25+75-u_kucuk*25))/2;
               WABL = u_kucuk*WABL_1 + (u_orta-u_kucuk)*WABL_2;
            }
            else{
               WABL_1 = ((1-c)*(0)+c*(75-u_orta*25+75))/2;
               WABL_2 = ((1-c)*(0)+c*(50-u_kucuk*25+50-u_orta*25))/2;
               WABL = u_orta*WABL_1 + (u_kucuk-u_orta)*WABL_2;
            }
         }
         else if(u_fazla>0 && u_orta>0 && u_kucuk>0 ){
            if(u_fazla>u_orta && u_orta>u_kucuk){
               WABL_1 = ((1-c)*(0)+c*(100+100))/2;
               WABL_2 = ((1-c)*(25+u_kucuk*25+25+u_orta*25)+c*(100+100))/2;
               WABL_3 = ((1-c)*(50+u_orta*25+50+u_fazla*25)+c*(100+100))/2;
               WABL = u_kucuk*WABL_1 + (u_orta-u_kucuk)*WABL_2 + (u_fazla-
u_orta)*WABL_3;
            }
            else if(u_kucuk>u_orta && u_orta>u_fazla){
               WABL_1 = ((1-c)*(0)+c*(100+100))/2;
               WABL_2 = ((1-c)*(0)+c*(75-u_orta*25+75-u_fazla*25))/2;
               WABL_3 = ((1-c)*(0)+c*(50-u_kucuk*25+50-u_orta*25))/2;
               WABL = u_fazla*WABL_1 + (u_orta-u_fazla)*WABL_2 + (u_kucuk-
u_orta)*WABL_3;
            }
            else if(u_orta>u_fazla && u_fazla>u_kucuk){
               WABL_1 = ((1-c)*(0)+c*(100+100))/2;
               WABL_2 = ((1-c)*(25+u_kucuk*25+25+u_fazla*25)+c*(100+100))/2;
               WABL_3 = ((1-c)*(25+u_fazla*25+25+u_orta*25)+c*(75-
u_orta*25+75-u_fazla*25))/2;
```

```
                WABL = u_kucuk*WABL_1 + (u_fazla-u_kucuk)*WABL_2 + (u_orta-
u_fazla)*WABL_3;
            }
            else if(u_orta>u_kucuk && u_kucuk>u_fazla){
                WABL_1 = ((1-c)*(0)+c*(100+100))/2;
                WABL_2 = ((1-c)*(0)+c*(75-u_kucuk*25+75-u_fazla*25))/2;
                WABL_3 = ((1-c)*(25+u_kucuk*25+25+u_orta*25)+c*(75-
u_orta*25+75-u_kucuk*25))/2;
                WABL = u_fazla*WABL_1 + (u_kucuk-u_fazla)*WABL_2 + (u_orta-
u_kucuk)*WABL_3;
            }
        }
        else{
            System.out.println("Şans değerlerini tekrar kontrol edin!!");
        }
        crisp_value[i]= WABL;
    }
  }
  private double ch_energy_level(double d){
    double E_ch = 0;
    E_ch = nodes * M * ((E_elec * Math.pow(10, -9)) + (e_fs * Math.pow(10,-15))
* d_BS + (E_DA * Math.pow(10,-9)));
    return E_ch;
  }
  private double node_energy_level(double d){
    double E_Tx = 0;
    if(d < d0){
      E_Tx = M * ((E_elec * Math.pow(10, -9)) + (e_fs * Math.pow(10,-15)) *
Math.pow(d,2));
    }
    else{
```

```
            E_Tx = M * ((E_elec * Math.pow(10, -9)) + (e_mp * Math.pow(10,-15)) *
Math.pow(d,4));
        }
        return E_Tx;
    }
    private void k_means_1(){
        distance_to_mean = new double[nodes+1][k+1];
        min_distance_to_mean = new double[nodes+1];
        mean_membership = new int[nodes+1];
        for(int i=1; i<=nodes; i++){
            for(int j=1; j<=k; j++){
                distance_to_mean[i][j] = euclidean_distance(coord[i][1],
m[j][1],coord[i][2],m[j][2]);
            }
        }
        for(int i=1; i<=nodes; i++){
            min_distance_to_mean[i] =  getMin(distance_to_mean[i]);
            for(int j=1; j<=k; j++){
                if(min_distance_to_mean[i] == distance_to_mean[i][j]){
                    mean_membership[i] = j;
                }
            }
        }
        double SSE_delta = SSE;
        SSE = 0;
        for(int i=1; i<=nodes; i++){
            SSE += Math.pow(min_distance_to_mean[i],2);
        }
        if(SSE==SSE_delta){
            adim_1();
        }
        else {
```

```
        k_means_2();
    }
}
private void k_means_2(){
    for(int i=1; i<=k; i++){
        double temp1 = 0;
        double temp2 = 0;
        int count = 0;
        for(int j=1; j<= nodes; j++){
            if(mean_membership[j]==i){
                temp1 += coord[j][1];
                temp2 += coord[j][2];
                count ++;
            }
        }
        m[i][1] = temp1/count;
        m[i][2] = temp2/count;
    }
    k_means_1();
}
private double euclidean_distance(double x1,double x2,double y1,double y2){
    return  Math.sqrt(Math.pow(x1-x2, 2)+ Math.pow(y1-y2, 2));
}
public static double getMin(double[] inputArray){
    double minValue = inputArray[1];
    for(int i=1;i < inputArray.length;i++){
      if(inputArray[i] < minValue){
         minValue = inputArray[i];
      }
    }
    return minValue;
}
```

```
    public static double getMax(double[] inputArray){
        double maxValue = inputArray[1];
        for(int i=1;i < inputArray.length;i++){
          if(inputArray[i] > maxValue){
             maxValue = inputArray[i];
          }
        }
        return maxValue;
    }
    private void set_chart() {
        series = new XYSeries[k+1];
        for(int i=1; i<=k; i++){
           series[i] = new XYSeries("Küme "+i);
        }
        ch_series = new XYSeries[k+1];
        for(int i=1; i<=k; i++){
           ch_series[i] = new XYSeries("Küme başı "+i);
        }
        dead_series = new XYSeries("Ölü Düğüm Noktası");
        /*------------------------------------------------------------ */
        this.chartPanel = createPanel();
        this.add(chartPanel, BorderLayout.CENTER);
        this.chartPanel.addChartMouseListener(this);
        CrosshairOverlay crosshairOverlay = new CrosshairOverlay();
        this.xCrosshair = new Crosshair(Double.NaN, Color.GRAY, new
BasicStroke(0f));
        this.xCrosshair.setLabelVisible(true);
        this.yCrosshair = new Crosshair(Double.NaN, Color.GRAY, new
BasicStroke(0f));
        this.yCrosshair.setLabelVisible(true);
        crosshairOverlay.addDomainCrosshair(this.xCrosshair);
        crosshairOverlay.addRangeCrosshair(this.yCrosshair);
```

```
        this.chartPanel.addOverlay(crosshairOverlay);
        /*------------------------------------------------------------ */
        JPanel control_1 = new JPanel();
        button = new JButton("Çalıştır");
        control_1.add(button);
        this.add(control_1, BorderLayout.SOUTH);
        /*------------------------------------------------------------ */
        JPanel control_2 = new JPanel();
        res_energy = new JLabel("");
        control_2.add(res_energy);
        this.add(control_2, BorderLayout.NORTH);
        //initialing swing timer
        timer = new javax.swing.Timer(100, getButtonAction());
        button.addActionListener((ActionEvent e) -> {
            if (!timer.isRunning()) {
                timer.start();
            } else {
                timer.stop();
            }
        });
    }
    private ActionListener getButtonAction() {
        ActionListener action = new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                int active_ch_count = coord_list.get(counter).length;
                button.setText(String.format("İterasyon sayısı: "+counter+",  Aktif düğüm
noktası: "+active_ch_count));
                if(counter>=iteration){
                  timer.stop();
                }
                else{
```

```
                moved.clear();
                cluster_heads.clear();
                for(int i=1; i<=k; i++){
                    series[i].clear();
                }
                for(int i=1; i<=k; i++){
                    ch_series[i].clear();
                }
                dead_series.clear();
                update(counter);
                counter++;
               // sleep(200);
             }
          }
       };
       return action;
    }
    private void update(int i) {
       for (int j = 0; j < head_list.get(i).length; j++) {
          ch_series[j+1].add(new XYDataItem(coord_list.get(i)[head_list.get(i)[j]-
1][1], coord_list.get(i)[head_list.get(i)[j]-1][2]));
       }
       for (int j = 0; j < ch_membership_list.get(i).length; j++) {
          series[ch_membership_list.get(i)[j]].add(new
XYDataItem(coord_list.get(i)[j][1], coord_list.get(i)[j][2]));
       }
       for (int j = 0; j < dead_coord_list.get(i).length; j++) {
          dead_series.add(new
XYDataItem((dead_coord_list.get(i))[j][1],(dead_coord_list.get(i))[j][2]));
       }
    }
    private ChartPanel createPanel() {
```

```
    JFreeChart jfreechart = ChartFactory.createScatterPlot(
       title, "X", "Y", createData(),
       PlotOrientation.VERTICAL, true, true, false);
    this.xyPlot = (XYPlot) jfreechart.getPlot();
    XYItemRenderer renderer = xyPlot.getRenderer();
    renderer.setSeriesPaint( 0 , Color.RED );
    renderer.setSeriesPaint( 1 , Color.GREEN );
    renderer.setSeriesPaint( 2 , Color.YELLOW );
    renderer.setSeriesPaint( 3 , Color.BLUE );
    return new ChartPanel(jfreechart){
       @Override
       public Dimension getPreferredSize() {
          return new Dimension(800, 600);
       }
    };
}
private XYDataset createData() {
    XYSeriesCollection dataset = new XYSeriesCollection();
    for(int i=1; i<=k; i++){
       dataset.addSeries(ch_series[i]);
    }
    for(int i=1; i<=k; i++){
       dataset.addSeries(series[i]);
    }
    dataset.addSeries(dead_series);
    return dataset;
}
@Override
public void chartMouseClicked(ChartMouseEvent e) {
    this.dataArea = this.chartPanel.getScreenDataArea();
    JFreeChart chart = e.getChart();
    this.xyPlot = (XYPlot) chart.getPlot();
```

```
        ValueAxis xAxis = this.xyPlot.getDomainAxis();
        double x = xAxis.java2DToValue(e.getTrigger().getX(), this.dataArea,
                RectangleEdge.BOTTOM);
        double y = xAxis.java2DToValue(e.getTrigger().getY(), this.dataArea,
                RectangleEdge.LEFT);
        /*---------------------------------------------------------- */
        for (int i = 0; i < coord_list.get(counter-1).length; i++) {
           double coord_node_x = coord_list.get(counter-1)[i][1];
           double coord_node_y = coord_list.get(counter-1)[i][2];
           if(Math.abs(x-coord_node_x)<1.5 && Math.abs(y-coord_node_y)<1.5){
              this.xCrosshair.setValue(coord_node_x);
              this.yCrosshair.setValue(coord_node_y);
              String energy_level = df.format(energy_list.get(counter-1)[i]);
               res_energy.setText("Kalan enerji seviyesi: " + energy_level+" J");
           }
        }
        for (int i = 0; i < dead_coord_list.get(counter-1).length; i++) {
           double coord_node_x = dead_coord_list.get(counter-1)[i][1];
           double coord_node_y = dead_coord_list.get(counter-1)[i][2];
           if(Math.abs(x-coord_node_x)<1.5 && Math.abs(y-coord_node_y)<1.5){
              this.xCrosshair.setValue(coord_node_x);
              this.yCrosshair.setValue(coord_node_y);
              String energy_level = df.format(dead_energy_list.get(counter-1)[i]);
              res_energy.setText("Kalan enerji seviyesi: " + energy_level+" J");
           }
        }
     }
  }
```