**DOKUZ EYLÜL ÜNİVERSİTESİ**
**FEN BİLİMLERİ ENSTİTÜSÜ**

# EN İYİ PROJE EKİBİNİN OLUŞTURULMASI İÇİN BULANIK MODELLER VE UYGULAMALARI

**Burak OKUR**

**Mayıs, 2017**
**İZMİR**

# EN İYİ PROJE EKİBİNİN OLUŞTURULMASI İÇİN BULANIK MODELLER VE UYGULAMALARI

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

**Burak OKUR**

**Mayıs 2017**
**İZMİR**

## YÜKSEK LİSANS TEZİ SINAV SONUÇ FORMU

**Burak OKUR,** tarafından **PROF. DR. EFENDİ NASİBOV** yönetiminde hazırlanan **"EN İYİ PROJE EKİBİNİN OLUŞTURULMASI İÇİN BULANIK MODELLER VE UYGULAMALARI"** başlıklı tez tarafımızdan okunmuş, kapsamı ve niteliği açısından bir Yüksek Lisans tezi olarak kabul edilmiştir.

.....

Prof. Dr. Efendi NASİBOV

Yönetici

.....

Doç. Dr. Müjgan Tuzen BEKİROĞLU

Jüri Üyesi

.....

Yrd. Doç. Dr. Arif GÜRSOY

Jüri Üyesi

Prof.Dr. Emine İlknur CÖCEN

Müdür

Fen Bilimleri Enstitüsü

## TEŞEKKÜR

Bu çalışmanın gerçekleştirilmesinde, iki yıl boyunca desteğini esirgemeyen, tecrübesi ile akademik hayatıma her şekilde katkıda bulunan, çok değerli danışmanım Prof. Dr. Efendi NASİBOĞLU 'a, çalışmalarım esnasında sabır gösterdiği ve her şekilde yanımda olduğu için eşim Merve'ye, çalışmamın içeriğini konferans bildirisi olarak hazırladıktan sonra İngilizce çevirisi kontrolleri konusunda yardımlarından dolayı kuzenim Emre OKUR ve Dokuz Eylül Üniversitesi Bilgisayar Bilimleri bölümünde Araştırma Görevlisi olan Alican DOĞAN 'a ve tüm hayatım boyunca her şekilde desteğini esirgemeyen babam, Hasan Özer OKUR 'a sonsuz teşekkürlerimi sunarım.

Bu tez, TUBITAK 115E194 nolu projesi ile kısmen desteklenmiştir.

Burak OKUR

# EN İYİ PROJE EKİBİNİN OLUŞTURULMASI İÇİN BULANIK MODELLER VE UYGULAMALARI

## ÖZ

Kutulama problemi, farklı hacimlere sahip objelerin sınırlı hacme sahip kutulara yerleştirilmesi ve kutu sayısının minimize edilmesi şeklinde ifade edilebilir. Günlük hayatta birçok uygulama alanı vardır. Program akışı oluşturma, kargo yükleme, bütçe planlama gibi örnekler gösterilebilir. Günümüze kadar problemin birçok kolu üzerinde çalışılmıştır. Bu kollar dahi başlı başına birer dal haline gelmiştir. Sırt çantası problemi, bulanık mantık ile kutulama bu dallara örnek gösterilebilir.

Bu tezde üzerinde çalışılan problem; oluşturulan takımlara kişilerin dağıtılması üzerine bir problemdir. Problemde kişilerin bir hacmi veya değerleri yoktur. Bunun yerine kişilerin takımlar ile bağımlılık ve uyumluluklarının derecelerini barındıran ilişki matrisleri vardır. Bu değerler sıfır ile bir kapalı aralığı içinde yer alan ondalık değerlerdir. Dolayısı ile problem klasik kutulama probleminden ayrılıp bulanık mantık içeren kutulama problemi sınıfına girmektedir. Burada, kişilerin takımlara dağılımından sonra oluşan çözüm kümesi, bulanık ilişki matrisleri kullanılarak dağılımın kalite derecesi hesaplanır. Takımların eleman sayıları da önceden tanımlanmış bir bulanık sayıya aidiyet değerleri ile derecelendirilir. Bu iki dereceden küçük olan değer, toplam kalite derecesidir. Bu kalite değeri maksimize edilmeye çalışılmıştır. Bu amaçla, çalışmada çeşitli çözüm algoritmaları geliştirilmiştir. Algoritma kodları CSharp programlama dilinde hazırlanmış ve hesaplama deneyleri yapılmıştır. Deneyler sonucu, Macar algoritmasına dayanan yaklaşımın daha etkin olduğu gözlenmiştir.

**Anahtar kelimeler**: Bulanık mantık, kutulama problemi, atama problemi, bulanık ilişki matrisleri.

# FUZZY MODELS TO DESIGN OPTIMAL PROJECT TEAM AND RELATED APPLICATIONS

## ABSTRACT

The problem of bin-packing can be expressed as placing objects with different volumes in boxes with limited volume and minimizing the number of boxes. There are many application areas in daily life. Creation of program flow, cargo packing and budget planning are some of examples. It has been worked on many variations of bin-packing problem. The knapsack problem, bin-packing with fuzzy logic are examples of them.

The problem studied in this thesis is; It is a problem on the distribution of the persons to the formed teams. The people in the problem do not have a volume or value. Rather, there are relational matrices that contain degrees of dependencies and compatibility of teams with teams. These values are decimal values between zero and one. Therefore, the problem is separated from the classic bin-packing problem and falls into the bin-packing problem which contains fuzzy logic. Here, the quality level of the distribution is calculated by using the fuzzy value matrices with the solution set formed after the distribution of the persons to the teams, and the element numbers of the teams are compared with the fuzzy number belonging values again. The value less than these two values is called the quality grade. This quality value has been tried to be maximized. For this purpose, various solution algorithms have been developed in the study. Algorithm codes are written in CSharp programming language and calculation experiments are done. Experiments have shown that the approach based on the Hungarian algorithm is more effective.

**Keywords**: Fuzzy logic, bin-packing problem, assignment problem, fuzzy relation matrices.

**İÇİNDEKİLER**

**Sayfa**

## ŞEKİLLER LİSTESİ

**Sayfa**

## TABLOLAR LİSTESİ

**SAYFA**

# BÖLÜM BİR
# GİRİŞ

Günümüzde birçok sorunu çözmek için ekip olarak çalışılmaktadır. Elde edilecek faydayı, gerek zaman gerekse sorunun çözümü olsun, maksimize etmek için takımın kaliteli olması gerekir. Çalışmada, kişiler ile ekipler arasındaki ilişki, uyum ve bağımlılık matrisleri ile ilişkilendirilmiştir. Takımların kapasitesi ise esnek olmasını sağlamak amacı ile bulanık sayı ile gösterilmiştir. Önerilen sezgisel algoritmalar ile küçük ölçekli problemlerde kaliteli dağımlar elde edilebilmişse de istikrarlı değildirler. Bu sonuçları daha istikrarlı hale getirmek için problem, Macar algoritması ile çalışabilecek hale getirilmiştir. Uygulanan testlerde, bu amacın başarıldığı gözlenmiştir.

Bu tezde, üzerinde çalışılan problem; oluşturulan takımlara kişilerin dağıtılması üzerine bir problemdir. Takımların oluşturulmasında, kişilerin takımlarla uyumları, kişilerin takımlarla bağlılıkları ve takım eleman sayılarının homojen oluşturulması kalite kriterleri olarak ele alınmıştır. Bu kalite derecelerinin hesaplanmasında bulanık ilişki matrisleri kullanılmıştır. Dolayısı ile problem klasik kutulama probleminden ayrılıp bulanık mantık içeren kutulama problemi sınıfına girmektedir. Bu bakımdan, tezin devamında kutulama problemi ve bulanık mantıkla ilgili bazı tanımları verilecektir.

Kaliteli proje takımı oluşturma probleminin çözümünde kişiler obje, takımlar da kutu olarak düşünüldüğünde kişilerin takımlara doldurulması problemi kutulama problemine benzetilip çözüm aranmıştır. Bu nedenle ilk başta kutulama problemi ve onun varyasyonlarından bahsedilecektir.

# BÖLÜM İKİ
# KUTULAMA PROBLEMİ

Kutulama problemi, farklı hacimlere sahip objelerin sınırlı hacme sahip kutulara yerleştirilmesi ve kutu sayısının minimize edilmesi şeklinde ifade edilebilir. Günlük hayatta birçok uygulama alanı vardır. Program akışı oluşturma, kargo yükleme, bütçe planlama gibi problemler, kutulama probleminin uygulama alanları için örnek gösterilebilir. Günümüze kadar problemin birçok varyasyonu üzerinde çalışılmıştır.

Boyutları verilen N adet obje ve kapasiteleri eşit olan kutular verilsin. Amaç, objeleri kutulara koyarken mümkün olan en az kutu kullanılmasıdır. Kutuların kapasitelerinin de aşılmaması gerekmektedir.

Kutuların kapasitesi 1 olsun. Problem şu şekilde formüle edilebilir; $a_1, a_2, \dots, a_n$ ; $0 \leq a_i \leq 1$ olmak üzere bir fonksiyon tanımlansın. $f: \{1, \dots, n\} \rightarrow \{1, \dots, k\}$ ve $f(i) = j$ ve $j \in \{1, \dots, k\}$ olmak üzere $k \in N$ sayısının minimize edilmesi hedeflenmektedir (Korte ve Vygen, 2006).

## 2.1 Sezgisel Çözüm Önerileri

Problemin çözümü için bazı açgözlü sezgisel algoritmalar önerilmiştir. Problem NP-Zor türünden olduğu için, tam polinomiyal-zamanlı bir çözümü yoktur. Bu bölümde bahsedilecek olan algoritmalar için bazı kısaltmalar kullanılmıştır. Next-Fit algoritması NF, Best-Fit algoritması BF ve First-Fit algoritması içinse FF olacak şekilde kısaltılmıştır. Aynı algoritmaların bir başka sürümünde, önce objeler boyutları azalacak şekilde sıralanıp sonra algoritma çalıştırılır. Bu algoritmalar next-fit decreasing (NFD), best-fit decreasing (BFD) ve first-fit decreasing (FFD) şekilde adlandırılmıştır. Algoritma içinde kullanılan ifadeler için; k, kutu sayısını, S o an kullanılan kutunun kapasitesini [0,1] aralığında değer ile gösterir, $S_k$ k. kutunun kapasitesini [0,1] aralığında değer ile gösterir. $f(i) = k$ ise i'ninci obje k'nıncı kutuya yerleştiğini göstermektedir. 1'den n'ye kadar olan objelerin boyutları $a_1, a_2, \dots, a_n$ şeklinde gösterilecektir.

### *2.1.1 Next-Fit Algoritması*

Obje kutuya sığıyorsa o kutuya yerleştirilir. Eğer kutuya yerleşmiyorsa yeni bir kutu alınır ve obje kutuya yerleştirilir. Algoritma şu şekilde ifade edilebilir;

$k = 1; S = 0;$
**for** $i = 1\ to\ n$ **begin**
    **if** $(S + a_i > 1)$ **begin**
        $k = k + 1;$
        $S = 0;$
    **endif**
    $f(i) = k;$
    $S = S + a_i;$
**endfor**.

**Teorem**: Algoritma O(n) zamanda çalışır. Herhangi bir I problemi için

$$NF(I) \leq 2\left(SUM(I)\right) - 1 \leq 2\,(OPT(I)) - 1 \tag{2.1}$$

eşitsizliği sağlanır (Korte ve Vygen, 2006).

### *2.1.2 First-Fit Algoritması*

Obje, ilk içine yerleşebildiği kutuya konur. Eğer yerleşebildiği bir kutu yoksa yeni bir kutuya yerleştirilir. Algoritması ise şu şekildedir.

k = 1; S = 0;
**for** $i = 1\ to\ n$ **begin**
    //en iyi kutu olarak yeni kutu işaretlenir
    $b = k + 1;$
    **for** $j = 1\ to\ k$ **begin**
        **if** $\left(S_j + a_i \leq 1\right)$ **begin**

```
                //içinde yer olan kutulardan objenin
                //sığdığı ilk kutu tercih edilir
                b = j;
                break;
            endif
        endfor
        // bulunan iyi kutu varsa ona yoksa yeni kutuya yerleştirilir.
        S_b = S_b + a_i;
        f(i) = b;
        if ( b == k + 1) k = k + 1;
endfor.
```

FF algoritma sonucu, NF algoritmasının sonucundan kötü olamaz. FF 2-faktor yaklaşım algoritmasıdır.

Garey ve Johnson tarafından kutulama probleminin herhangi bir I örneği için FF algoritmasıyla bulunan yaklaşık çözümün aşağıdaki koşulu sağladığı kanıtlanmıştır (Garey ve Jonhson, 1981):

$$FF(I) \leq \left\lceil \frac{17}{10} OPT(I) \right\rceil. \quad (2.2)$$

Benzer şekilde (Simchi-Levi, 1994) daha iyi olan şu eşitsizliği sunmuştur:

$$FF(I) \leq \frac{7}{4} OPT(I). \quad (2.3)$$

### *2.1.3 Best-Fit Algoritması*

Obje hali hazırda bulunan ve içine girebileceği kutular arasında içine girdiğinde en az boşluk bıraktığı kutuya yerleştirilir. Eğer yerleşebileceği bir kutu yoksa yeni bir kutu alınır ve obje o kutuya yerleştirilir. Algoritma şu şekilde gösterilir;

```
k = 1; S = 0; i = 0;
```

**for** $i = 1\ to\ n$ **begin**

$b = k + 1$; //en iyi kutu olarak yeni kutu işaretlenir

w = 1; //en iyi kutunun boyutu verilir.

**for** $j = b\ to\ 1$ **begin**

**if** $\left(S_j + a_i \leq 1\ and\ S_j + a_i < w\right)$ **begin**

//önceki adımlarda oluşturulan

//kutular içinde daha uygun olan tercih edilir

b = j;

w = $S_j + a_i$;

**endif**

**endfor**

// bulunan iyi kutu varsa ona yoksa yeni kutuya yerleştirilir.

$S_b = S_b + a_i$;

f(i) = b;

**if** ( $b == k + 1$) $k = k + 1$;

**endfor**.

### *2.1.4 Azalan Sıralı Algoritmalar*

Verilen algoritmalar uygulanmadan önce boyutlarına göre azalacak şekilde sıralanır daha sonra hangi algoritma seçildiyse sıralı objeler kullanılarak algoritma yürütülür.

*Teorem*: FFD algoritması, kutulama probleminin $\frac{3}{2}$ faktör yaklaşık algoritmasıdır (Simchi-Levi, 1994).

*Teorem*: Örneği I olan bir kutulama problemleri için, (Minyi, 1990)

$$FFD(I) \leq \frac{11}{9} OPT(I) + 1. \quad (2.4)$$

## 2.2 2D Kutulama Problemi

Kutulama probleminin varyasyonlarından biri de 2-boyutlu kutulama problemidir. Objelerin en ve boy olacak şekilde iki adet boyutu vardır. Kutuları da yine 2-boyutlu

levhalar olarak düşünebilir. Objeler bu kutulara en az sayıda kutu kullanılması şartı ile yerleştirilecektir. Yerleştirme esnasında objeler ile kutuların kenarlarının paralel olması, objelerin üst üste gelmemesi ve objelerin döndürülmemesi gerekmektedir.

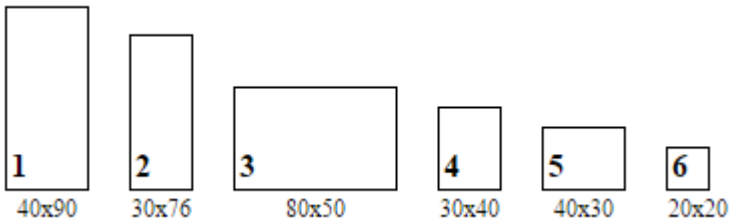
Birçok uygulama alanı bulunmaktadır. Endüstriyel anlamda mobilyalar için plaka halinde bulunan bloklardan parçalar elde edilirken yaygın şekilde kullanılır. Gazete sayfalarına haberleri yerleştirmek yine bir başka uygulamasıdır. Bazı yeni kısıtlamalar kullanılarak yeni uygulamalarda elde edilebilir. Örnek olarak gazete dizaynı verilebilir; gazete sayfa sayısı ve haber sayısı sabit olsun, bazı haberlerin belli sayfalarda yayınlanması durumu dikkate alınarak en büyük boşluklar elde edilir. Elde edilen boşluklara yerleştirilen reklamlar ile gelirin maksimize edilmesi hedeflenir.

Özel durum olarak objelerin ve kutuların boyutlarından ilkini sabit bir değer alırsak problem kutulama problemine çevrilir. Kutulama probleminin NP-zor olduğundan 2-boyutlu kutulama problemi de NP-zordur.

Literatürdeki çoğu offline algoritmalar açgözlü ve sezgisel algoritmalardır ve iki sınıfta sınıflandırılabilirler (Lodi, Martello ve Vigo, 2002).

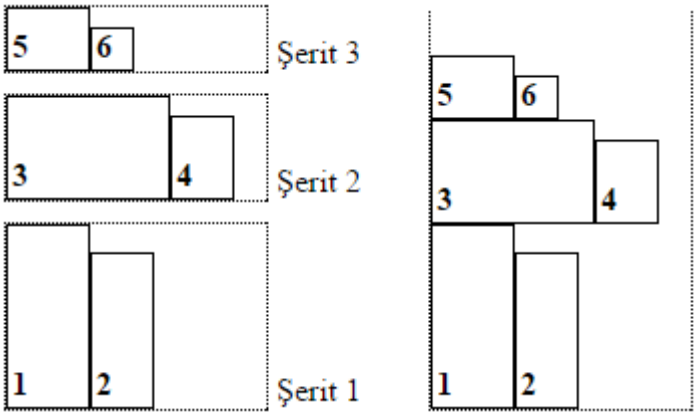
- 1 aşamalı algoritmalar: objeler direkt kutulara yerleştirilir.
- 2 aşamalı algoritmalar: objeler önce, ilk boyutları baz alınarak bir şerite yerleştirilir. Bu şeritlerin bir boyutu sınırlı diğer boyutu ise bu şerite yerleştirilen en büyük boyutlu elemanın ikinci boyutu kabul edilir. İkinci olarak ise bu şeritler, elde edilen ikinci boyutlarına göre kutulara yerleştirilerek, algoritma tamamlanır.

FFD, BFD, NFD algoritmalarını Şekil 2.1'deki objeler kullanılarak gösterilecektir.

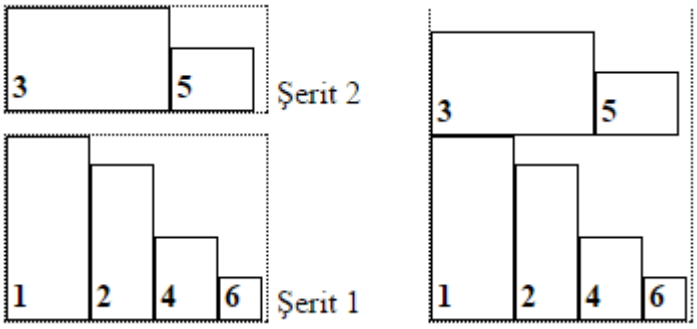



Şekil 2.1 Algoritmalarda kullanılacak objeler

Şekil 2.2'de next-fit descreasing algoritması ile 2 aşamalı çözüm yapılmış. Resimde sol tarafta 1. aşamada oluşturulan şeritler görülmektedir. 2. Aşamada ise bu şeritler birer obje olarak ele alınıp kutuya yerleştirilmiştir. Şekil 2.3'te sırası ile first-fit algoritması kullanılan 2 aşamalı çözüm görülmektedir.




Şekil 2.2 NFD algoritması ile çözüm




Şekil 2.3 FFD algoritması ile çözüm

Bu algoritmalar dışında başka sezgisel algoritmalarda zamanla ortaya atılmıştır. *Floor-Ceiling* algoritması, sol alt kısımdan başlayıp kutuya yerleştirilen objeler tamamlandığında bu sefer sağ üst kısımdan sola doğru sığan objelerin konması ile devam eder. *Left-Bottom* algoritmasında ise kutunun sol alt kısmından başlanarak objeler yerleştirilir. Son yerleştirilen objenin sağından devam edilir. Yer kalmadığında, boş kalan kısmın sol altından başlanılarak aynı adımlar tekrar edilir.

2-boyutlu kutulama probleminin çeşitleri de vardır. Günlük hayatta her zaman kenarlara paralel olmama durumu ya da obje veya kutuların dikdörtgen olmama durumunu içeren problemlerle karşılaşılır. Mesela gazete problemi çözümünde

objeleri döndüremezsiniz. Ancak bir MDF plakadan mobilya için parça kesiyorsanız kesilen parçanın oryantasyonu önemsizdir. 90 derecelik dönme ile verimli çözümlere ulaşılabilir.

## 2.3 Diğer Bazı Çalışmalar

Kutulama probleminin NP-zor olduğu bilinmektedir. Polinomiyal zamanda bir çözüm bulmak, şu an için mümkün değildir. Fakat yapılan çalışmalar birçok iyileştirme ve hızlandırma içermektedir.

Örneğin; Optimal kutu paketlemesi için sunulan bir çözüm önerisi; objelerin farklı kutulara nasıl yerleşebileceğini araştırmak yerine, kutuların içeriğinin nasıl farklı olabileceğini araştırmıştır (Korf, 2002). Yeni algoritma, bulunan en iyi kutulama algoritmalarından daha hızlı çalışmış hatta 60 objelik problem kümelerinde 1000 kata kadar hız iyileşmesi olduğu söylenmiştir. Aynı çalışmada 2 yenilik daha yapılmıştır, algoritmada kullanılan sabit faktör değiştirilmiş ve arama ağacındaki gereksiz arama kısımları temizlenmiştir (Korf, 2003).

2005 yılında *Clique-Graph* ile kutulama problemi üzerine çalışma yapılmış ve bazı çözüm önerileri sunulmuştur. Bu graf yardımı ile çelişkili objeler işaretlenir ve problem çözülürken bu objelerin aynı kutuya yerleştirilmemesi gerekmektedir (McCloskey ve Shankar, 2005).

Sırt çantasına dayalı akıl yürütmeyi içeren yayılım kuralları ve gereken kutu sayısının alt sınırını kullanan bir kısıt yardımıyla, kutulama probleminin çözümü için yapılan arama sayısı önemli ölçüde azaltılmıştır. Standart bir kutu arama stratejisi ile birleştiğinde, kutulama algoritmalarına önemli bir rakip olabilmektedir. (Shaw, 2004)

2-boyutlu kutulama probleminin çözümü için (Chazelle, 1983) tarafından alt-sol sezgisel yaklaşımı üzerine efektif bir çözüm sunulmuştur. Giyotin 2-boyutlu kutulama üzerine yapılan çalışmada elde edilen sonuç bir asimptotik polinom zaman yakınlık şemasıdır (Bansal, Lodi ve Sviridenko, 2005).

3-boyutlu kutulama problemi endüstriyel anlamda birçok alanda kullanılmaktadır ve bilgisayar bilimi ile uğraşan kimseler için pratik ve efektif bir çözüm oluşturmak zorlu bir mecradır. Bu konuda kutuları daha küçük kutulara parçalayıp o parçaları doldurma ve boş alanı minimize etmeyi hedefleyen "*Peak filling slice push (PFSP)*" isimli yöntem, testlerde önemli bir performans sergilemiştir (Maarouf, Barbar ve Owayjan, 2008). Bir başka çalışmada ise paketlenen objelerin, 3 boyutlu kutuların yüzeyleri paralel olacak şekilde 6 yönde dönmesine izin verilmiş ve çözümü üzerine çalışılmıştır (Wu, Li, Goh ve Souza, 2011).

2-boyutlu kutulama problemi üzerine Jukka Jylänki yaptığı detaylı bir araştırmada "*maxrects*" algoritması en iyi sonuçları veren algoritma olarak bulunmuştur. Eğer kutular her seferinde bir defa olacak şekilde işlenecekse "*skyline*" algoritması daha iyi sonuç vermektedir. "*guillotine*" algoritması "*maxrect*" algoritmasından daha hızlı bulunmuş fakat verimi biraz geride kalmıştır. "*shelf*"' algoritması ise kolay adapte olması ile ön plana çıktığı ifade edilmiştir (Jylänki, 2010).

Kutulama probleminin çevrimiçi versiyonları üzerine de birçok çalışma yapılmıştır (György, Lugosi ve Ottucsak, 2010). Bulut sunucular üzerinde işleri dağıtan, dağılan işlerin işlemciler üzerinde en kısa süre durmasını hedefleyen çalışmalar vardır (Tang, Li, Ren ve Cai, 2016).

# BÖLÜM ÜÇ
# BULANIK MANTIK

Takımlar oluşturulurken dikkat edilecek en önemli nokta kişiler ve takımlar arasındaki ilişkilere uygun şekilde dağılım yapılmasıdır. Bu dağılım yapılırken takımdaki kişilerin sayıları da dağılımın kalitesini etkilemektedir. Bu ilişkiler ve eleman sayılarının değerlendirilmesi için bulanık mantık kullanılmıştır.

## 3.1 Tanımı

Klasik kümeler kesin sınırlara sahiptir. Örneğin; 180'den büyük reel sayıları;

$$A = \{ x \,|x > 180 \; ve \; x \in \mathbb{R} \} \tag{3.1}$$

şeklinde gösterilir. Burada görüldüğü üzere kesin sınırlar mevcuttur. Bu sınırlar içindeki $x$'ler bu kümeye ait iken sınırlar dışında kalanlar ise kümeye ait değildirler. Bu klasik kümelerin birçok kullanım alanları vardır ve matematik alanında çok önemli bir araç olduğu bellidir. Ancak klasik kümeler her zaman insan doğasını yansıtmazlar. Çünkü gerçekte durum her zaman bu kadar net değildir. Örneğin bu A kümesi, boyu uzun insanlar kümesini ifade etsin. X ise cm cinsinden insanların boyları olsun. Eğer bir insan 180 cm'den uzunsa ifade edilen kümeye göre uzun bir insan olacaktır. Ancak bu doğal bir gösterim değildir. Eğer bir insanın boyu 180 cm ise o zaman bu insan A kümesine göre uzun boylu sayılmayacaktır. 181 cm boya sahip birinin uzun, 180 cm boya sahip birinin kısa olması doğal olmayan çok keskin bir geçiştir.

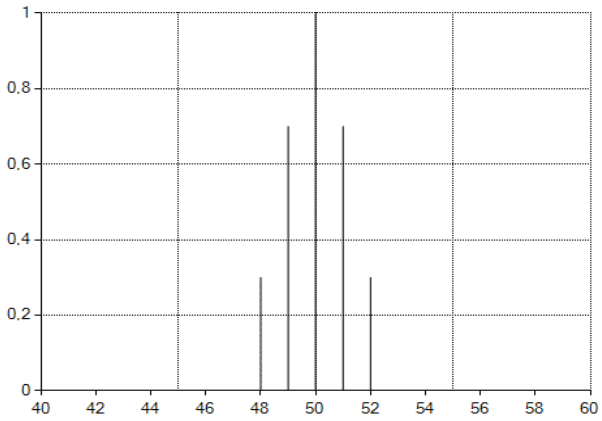
X, genel olarak x ile gösterilen objelerin bir kümesi olsun. X' deki bir bulanık kümeye de A diyelim ve bu küme sıralı çiftlerden oluşan bir küme olsun. A şu şekilde gösterilir;

$$A = \{(x, \mu_A(x)) | x \in X \}. \tag{3.2}$$

Burada $\mu_A(x)$ elemanın kümeye aitlik fonksiyonu (kısaca AF) olarak isimlendirilir. AF fonksiyonunun görüntü kümesi [0,1] aralığındadır. Örneğin, 50 yaş civarı kimseler şu şekilde gösterilebilir;

$$A=\{(48,0.3),(49,0.7),(50,1),(51,0.7),(52,0.3)\} \quad (3.3)$$

Şekil 3.1'de kesikli bir biçimde verilmiş bir bulanık küme grafiği verilmiştir.

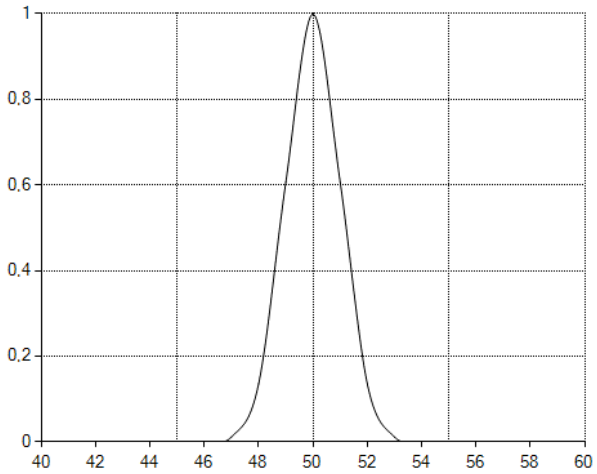



Şekil 3.1 İkililer şeklinde verilmiş bulanık küme

50 yaş civarı kimseleri içeren kümeyi üyelik fonksiyonu ile devamlı bir biçimde ifade edelim, kümenin adı B olsun;

$$B = \left\{(x, \mu_x(x)) \mid \mu_x(x) = \left(x, e^{-\left(\frac{(x-50)^2}{2}\right)}\right)\right\} \quad (3.4)$$

Şekil 3.2'de ise 50 yaş civarı kişiler devamlı şekilde ifade edilmiştir.




Şekil 3.2 Üyelik fonksiyonu ile devamlı bulanık küme

Bir diğer örnek de şu şekilde olabilir. Türkiye'nin en kalabalık şehirleri kümesine A kümesi diyelim.

$$A = \{\, \text{İ}zmir, \text{İ}stanbul, Ankara \,\} \tag{3.5}$$

A kümesinde İzmir, İstanbul ve Ankara şehirleri kalabalık olarak nitelenmiş fakat İstanbul'un nüfusu İzmir ve Ankara'nın toplan nüfusundan daha fazladır. Bu bilginin bir fark ortaya koyması için bu elemanlar kümeye aidiyet değeri ile verilir. Bu bulanık küme B ile gösterilsin;

$$B = \{\, (\text{İ}zmir, 0.3), (\text{İ}stanbul, 1), (Ankara, 0.4)\} \tag{3.6}$$

Kümedeki gösterimden dahi İstanbul'un diğer şehirlerden çok daha kalabalık olduğu gözlenebilmektedir. Bir karar alınırken, bu küme kullanıldığında daha sağlıklı analizler yapılabilir.

Bulanık kümelerin farklı gösterimleri olabilmektedir. Aynı B kümesi şu şekilde de gösterilebilir;

$$B = 0.3/\text{İ}zmir + 1/\text{İ}stanbul + 0.4/Ankara \tag{3.7}$$

Bulanık kümeler ile aritmetik işlemler yapılabilmektedir. Aynı zamanda bu kümelere birleşim ve kesişim gibi operatörler de uygulanabilmektedir (Zadeh, 1965).

## 3.2 Parametreler ile Bir Boyutlu Üyelik Fonksiyonları

Bulanık kümeler içinde tanımlanan bir boyutlu üyelik fonksiyonları için birkaç standart fonksiyon tanımlanmıştır. Bu fonksiyonlar, kümenin elemanına bağlı bir tane, fonksiyon türüne göre üç veya dört sabit parametre ile ifade edilir. Tez içinde yamuk şekille gösterilen üyelik fonksiyonu kullanılmıştır ancak herhangi bir üyelik fonksiyonu isteğe göre tercih edilebilir.

### *3.2.1 Üçgensel Üyelik Fonksiyonu*

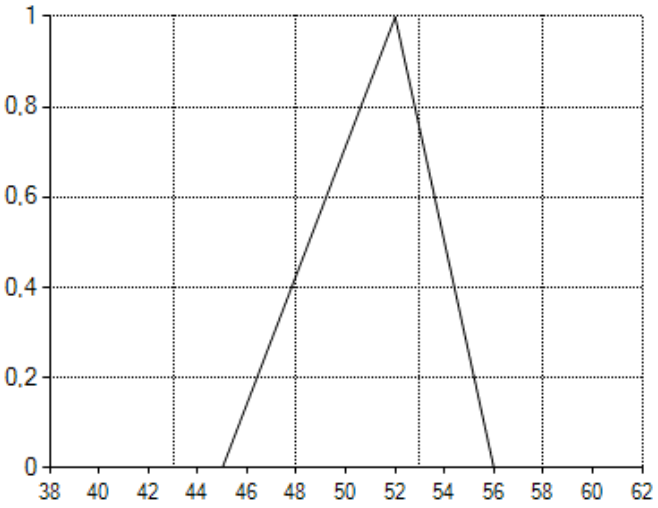
AF $\{a, b, c\}$ olacak şekilde 3 parametre ile ifade edilir. Üçgenin sol ayağı 'a', tepe noktası 'b' ve sağ ayağı ise 'c' ile ifade edilecektir. Dolayısı ile $a \leq b \leq c$ ilişkisi sağlanmalıdır.

$$\text{üç}gensel(x; a, b, c) = \begin{cases} 0\,, & x \leq a, \\ \frac{x-a}{b-a}\,, & a \leq x \leq b, \\ \frac{c-x}{c-b}\,, & b \leq x \leq c, \\ 0\,, & c \leq x \end{cases} \tag{3.8}$$

Üçgensel üyelik fonksiyonu parçalı fonksiyon şeklinde ifade edilmiştir. Alternatif olarak;

$$\text{üç}gensel(x; a, b, c) = \max\left(\min\left(\frac{x-a}{b-a}, \frac{c-x}{c-b}\right), 0\right) \tag{3.9}$$

şeklinde max-min kompozisyonu olarak da ifade edilebilir. Şekil 3.3'te üç$gensel(x; 45{,}52{,}56)$ parametreleri ile belirlenen bir üçgensel üyelik fonksiyon grafiği gösterilmiştir.




Şekil 3.3 Üçgensel üyelik fonksiyonu örneği

### *3.2.2 Yamuksal Üyelik Fonksiyonu*

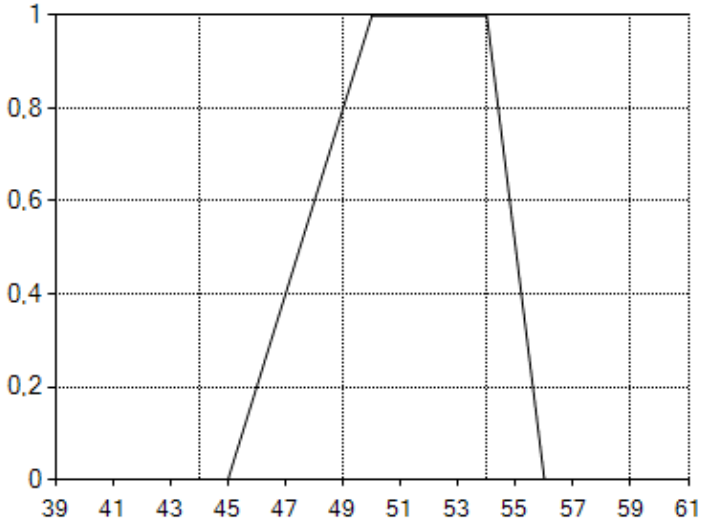
AF $\{a, b, c, d\}$ olacak şekilde 4 parametre ile ifade edilir. Şekil 3.4'te de görüldüğü gibi yamuğun sol ayağı 'a', sol tepe noktası 'b', sağ tepe noktası 'c' ve sağ ayağı ise 'd' notasyonları ile ifade edilir. Burada da yine $a \leq b \leq c \leq d$ ilişkisi sağlanmalıdır. Özel olarak, $b = c$ sağlandığı durumda üçgensel üyelik fonksiyonu ile aynı işleve sahip olur.

$$yamuksal(x; a, b, c, d) = \begin{cases} 0\,, & x \leq a. \\ \frac{x-a}{b-a}\,, & a \leq x \leq b. \\ 1\,, & b \leq x \leq c. \\ \frac{d-x}{d-c}\,, & c \leq x \leq d. \\ 0\,, & d \leq x \end{cases} \tag{3.10}$$

olacak şekilde parçalı fonksiyon olarak ifade edilebilir. Alternatif olarak;

$$yamuksal(x; a, b, c, d) = \max\left(\min\left(\frac{x-a}{b-a}, 1, \frac{d-x}{d-c}\right), 0\right). \tag{3.11}$$

şeklinde verilebilir bu ise max, min kompozisyonudur.

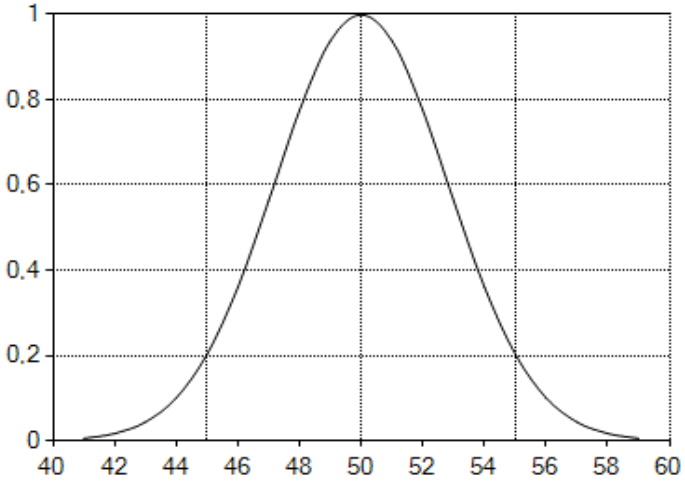



Şekil 3.4 Yamuksal üyelik fonksiyonu örneği

### *3.2.3 Gaussion Üyelik Fonksiyonu*

Gaussion fonksiyonu $\{c, d\}$ olacak şekilde 2 parametreden oluşur. 'c' fonksiyonun merkezini ifade ederken, 'd' ise fonksiyonun genişliğini belirler. Gaussion fonksiyonu Şekil 3.5'de görüldüğü gibidir.

$$gaussion(x; c, d) = e^{-\frac{1}{2}\left(\frac{x-c}{d}\right)^2}. \tag{3.12}$$



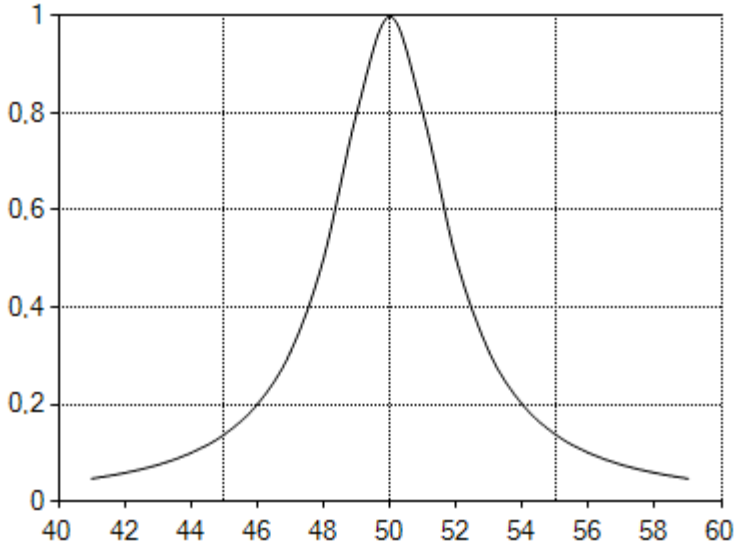

Şekil 3.5 Gaussion fonksiyon örneği

### *3.2.4 Genelleştirilmiş 'Bell' Üyelik Fonksiyonu*

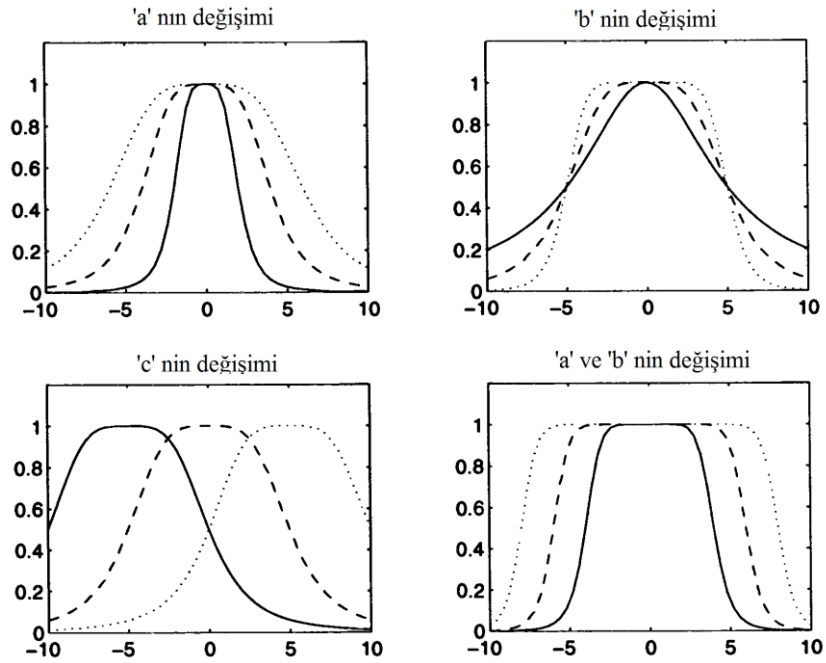
Genelleştirilmiş '*Bell*' fonksiyonunu ifade etmek için $\{a, b, c\}$ olmak üzere 3 parametreye ihtiyaç vardır. Fonksiyon içinde kullanılan 'c' parametresinin değişmesi fonksiyon merkezini değiştirir. 'a' parametresinin değişmesi fonksiyonun ayaklarının değişmesine olanak sağlar. 'b' parametresi ise fonksiyonun tepesinin yayılması veya daralması üzerine etkilidir. Şekil 3.7 üzerinde parametrelerin etkileri detaylı gösterilmiştir.

$$bell(x; a, b, c) = \frac{1}{1+\left|\frac{x-c}{a}\right|^{2b}} \tag{3.13}$$

$bell(x; 2{,}1{,}50)$ parametreleri ile '*Bell'* fonksiyonu Şekil 3.6'de gösterilmiştir.




Şekil 3.6 Bell fonksiyon örneği

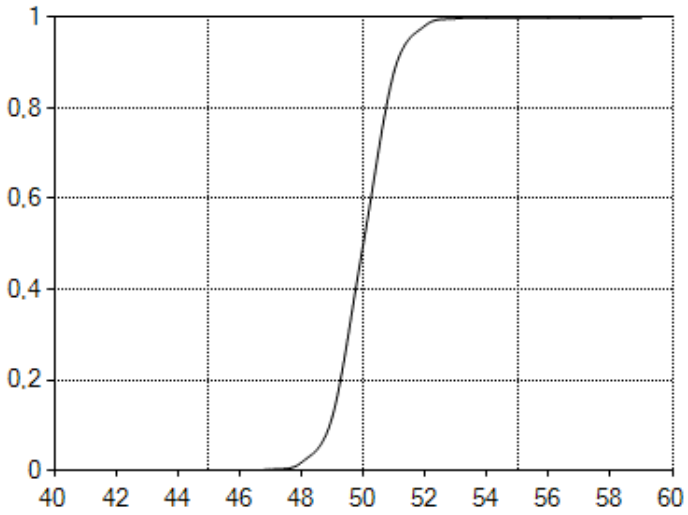



Şekil 3.7 Parametrelerin değişiminin "bell" fonksiyonu üzerindeki etkileri

### *3.2.5 Sigmoid Üyelik Fonksiyonu*

Sigmoid üyelik fonksiyonu için $\{a, c\}$ olmak üzere 2 parametre kullanılır. $a = 2$ ve $c = 50$ için sigmoid fonksiyonunun grafiği Şekil 3.8'de verilmiştir.

$$sig(x; a, c) = \frac{1}{1+e^{[-a(x-c)]}}. \tag{3.14}$$



Şekil 3.8 Sigmoid üyelik fonksiyonu

### 3.3 Bulanık İlişkiler

İkili bulanık ilişkiler, $X\ x\ Y$ olacak şekilde X kümesindeki her eleman ile Y kümesinde ki her elemanın [0,1] aralığındaki derecelerden ilişkilerini barındıran bulanık kümelerdir. Tekli bulanık ilişkiler ise 1 boyutlu bir üyelik fonksiyonudur. İkili bulanık ilişkileri 2 boyutlu bir üyelik fonksiyonu şeklinde tanımlayabiliriz ve bu şekilde devam eder. Bulanık ilişki uygulamaları bulanık kontrol veya karar verme mekanizmaları gibi alanlarda kullanılabilir.

X ve Y iki küme olsun.

$$R = \{ \big((x,y), \mu_R(x,y)\big) \,\big|\, (x,y) \in X \times Y \} \tag{3.15}$$

şeklinde tanımlanan bir ikili bulanık ilişkidir. Burada $\mu(x,y)$ iki boyutlu bir üyelik fonksiyonudur. R aynı zamanda matris olarak şu şekilde gösterilir;

$$R = \begin{bmatrix} r_{11} & \cdots & r_{1\mathrm{m}} \\ \vdots & \ddots & \vdots \\ r_{\mathrm{n}1} & \cdots & r_{\mathrm{nm}} \end{bmatrix},\ i = 1, \dots, \mathrm{n};\ j = 1, \dots, \mathrm{m},;\ r_{\mathrm{ij}} = \ \mu\big(x_i \in X, y_j \in Y\big) \tag{3.16}$$

$R_1 ve\ R_2$ adında iki tane bulanık ilişki matrisi $X \times Y\ ve\ Y \times Z$ üzerine tanımlı olsun. $R_1\ ve\ R_2$ matrisi arasındaki max-min kompozisyonu şu şekilde ifade edilmiştir.

$$R_1 \circ R_2 = \left\{ \left[ (x,z), \max_y \min\big(\mu_{R_1(x,y)}, \mu_{R_2(y,z)}\big) \right] \mid x \in X, y \in Y, z \in Z \right\}. \tag{3.17}$$

yada benzer şekilde,

$$\mu_{R_1 \circ R_2}(x,z) = \max_y \min\big(\mu_{R_1(x,y)}, \mu_{R_2(y,z)}\big) \tag{3.18}$$

$$= \vee_y \left[\mu_{R_1(x,y)} \wedge \mu_{R_2(y,z)}\right], \tag{3.19}$$

burada $\wedge\ ve\ \vee$ karakterleri sırası ile min ve max fonksiyonunu temsil etmektedir.

### *3.3.1 İlişki Matrislerinin Özellikleri*

X bir bulanık küme, I birim matris, $R(x,y)$ bir bulanık ilişki ve $\forall a,b,c \in X$ için;

***Yansıma özelliği***: $\forall a: R(a,a) = 1$ şartı sağlanıyorsa $R$ bulanık ilişkisinin yansıma özelliği vardır. Eğer $\forall a: R(a,a) \geq \epsilon$ şartını sağlıyorsa bu durumda $R$ bulanık ilişkisi $\epsilon$ dereceden yansıma özelliğine sahiptir denir.

***Simetri özelliği***: $\forall a,b: R(a,b) = R(b,a)$ durumu var ise $R$ bulanık ilişkisi simetri özelliğine sahiptir.

***Geçişlilik***: $R(a,c) \geq \min\{R(a,b), R(b,c)\}$ özelliği sağlanıyorsa $R$ geçişlilik özelliğine sahiptir (Pedrycz ve Gomide, 2007).

### *3.3.2 Kapalı Geçişlilik*

X sınırlı bir küme olsun. X kümesi üzerinde tanımlı bulanık ilişkiler içinde $\tilde{R}$ ile ifade edilen benzersiz bir bulanık ilişki vardır. $\tilde{R}$ hem R'yi içerir hem de R'nin içerdiği tüm bulanık ilişkileri içerir (Baets ve Meyer, 2003). Buna R'nin kapalı geçişliliği denir. X'in n elemanı olduğunu kabul edersek;

$$\tilde{R} = \text{trans(R)} = \text{R} \cup \text{R}^2 \cup \ldots \cup \text{R}^\text{n} \tag{3.20}$$

şeklide tanımlanabilir (Pedrycz ve Gomide, 2007). Burada;

$$R^2 = R \circ R, \ldots, R^p = R \circ R^{p-1} \tag{3.21}$$

$$R \circ R(x,y) = \max_z \min\{R(x,z), R(z,y)\} \tag{3.22}$$

işlemleri kullanılmaktadır. Eğer R ilişkisi 'yansıma' özelliğine sahipse $\cup$ operatörüne gerek kalmaz.

$$I \subseteq R \subseteq R^2 \subseteq \cdots \subseteq R^{n-1} = R^n \tag{3.23}$$

$R$ bulanık ilişkisinin kapalı geçişliliğini $R^n = R^{n-1}$ olana kadar max-min çarpımı yapılarak hesaplanabilir. Bu işlemin zaman karmaşıklığı $O(n^3 \log_2 n)$, bellek (space) karmaşıklığı ise $O(n^2)$'dir (Naessens, Meyer ve Baets, 2002). Algoritma şu şekildedir;

**while**( **true** )
    $\tilde{R} = R \cup (R \circ R)$;
    **if** ( $\tilde{R}$ == R ) **return** $\tilde{R}$;
    $R = \tilde{R}$;
**endwhile**.

Bir başka algoritma, geçişli kapalı matrisi $O(n^3)$ zaman karmaşıklığında ve $O(n^2)$ yer karmaşıklığında bulmaktadır (Kandel ve Yelowitz, 1974). Bu algoritma, en kısa yol algoritması olan floyd-warshall algoritmasının bir modifikasyonudur (Rardin, 1998). Floyd-warshall algoritması $R = \begin{bmatrix} r_{11} & \cdots & r_{1n} \\ \vdots & \ddots & \vdots \\ r_{n1} & \cdots & r_{nn} \end{bmatrix}$ şeklindeki kare ilişki matrisinden kapalı geçişli matrisi şu şekilde bulmaktadır;

$R_{ij}$; R ilişki marisi;
**for** $k = 1$ *to* $n$ **begin**
    **for** $i = 1$ *to* $n$ **begin**
        **for** $j = 1$ *to* $n$ **begin**
            $R_{ij} = \max\{R_{ij},\ \min\{R_{ik}, R_{kj}\}\}$;
        **endfor**
    **endfor**
**endfor**.

Benzer karmaşıklıklara sahip alternatif algoritmalar için ayrıntılı inceleme (Naessens, Meyer ve Baets, 2002) makalesinde bulunabilir. Çok daha büyük sayılı matrisler, ayrık matrisler ile ifade edilmektedir. Bulanık ilişkiler, ayrık matrislerde ifade edildiğinde ise ayrık matrisler ile çözüme uygun algoritmalar gerekmektedir.

Zaman karmaşıklığı ortalama $O(n\ log^4 n)$ olan algoritmalar (*Incremental transitive closure algoritması*) geliştirilmiştir (Wallace, 2006).

### *3.3.3 İlişki Matrislerinin Kullanım Alanları*

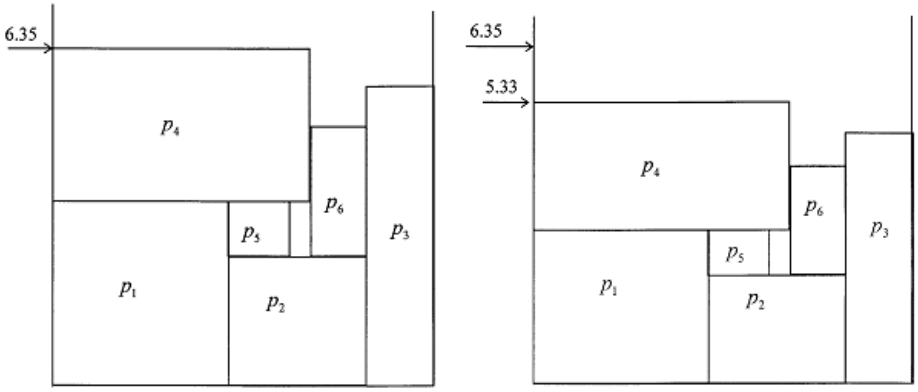
İlişki matrislerinin kullanım alanları yaygındır. Kutulama problemi dahilinde bir obje ile kutunun arasındaki ilişki değerlendirilebileceği gibi (Nasibov, 2004; Nasibov ve diğer., 2004; Nasibov ve Kınay, 2006) , farklı iki objenin arasındaki ilişkiyi de ifade edebilmektedir (Nasibov, 2004). Kutulama dışında bir kaç giysinin beden ölçüleri, bir kişinin beden ölçüleri ile bulanık bir sayı yardımı ile bir ilişki matrisinde tanımlanıp, kişiye en uygun bedeni bulma gibi çalışmalarda da kullanılmıştır (Nasibov ve diğer., 2016; Vahaplar ve diğer., 2016). Kapalı geçişlilik matrisini hızlı bir şekilde bularak alternatif çözüm önerileri arasından sıyrılan bir çalışmada bulanık kümeleme üzerinedir (Nasibov ve diğer., 2015).

# BÖLÜM DÖRT
# BULANIK MANTIK İLE KUTULAMA PROBLEMİ

Bulanık mantık ile kutulama problemi birçok farklı şekilde ele alınmıştır. Kimi zaman objelerin boyutları bulanık sayı olarak düşünülmüş, kimi zaman aynı kutuya konulacak objelerin arasında bulanık bir ilişki tanımlanmış, kimi zaman da obje ile kutu arasındaki bulanık ilişki ifade edilmiş ve yerleşimin kalitesine dikkat çekilmiştir. Düşünüldüğünde gerçek hayatta birçok uygulaması olabileceği görülmektedir. Mesela bir kargo şirketi, kırılacak eşyalar ile diğer eşyaları aynı kutuda taşımaması veya herhangi bir eşyanın kutunun belirli bir yerinde taşınması gibi birçok seçeneği değerlendirmek isteyebilir. Yazılan uygulama, okutulan etiketlerin ardından eşyaları yüklenilen kutuya önce kırılmaz objeleri daha sonra hassas objeleri yerleştirecek şekilde yönlendirebilir veya kısıtlarına göre aynı kutuda farklı bölümlere koymak yerine tamamen farklı kutulara yönlendirebilir.

## 4.1 Kim, Kwang ve Yoo Yaklaşımı

Bu çalışmada, kutulama problemi ile çözülen kumaş toplarından en uygun kumaş kesimi problemini, kumaş parçaların boylarını bulanık sayı olarak alınmış, maliyet ve memnuniyet arasındaki ilişkiye göre bu boylar hesaplanmıştır (Kim, Kwang ve Yoo, 2001). Burada kesilen kumaşların yüksekliği bulanık bir sayı olarak ifade edilmiştir. Kumaşta kısaltmaya gitmek, maliyet olarak anlık avantaj sağlasa da uzun vadede bir takım dezavantajları olabilecektir. Örneğin tüketicinin memnuniyetsizliği o tüketiciyi o üründen uzaklaştırabilir veya geri iadeler olabilir. Markaya olan güvenin azalmasına kadar gidebilecek ciddi sonuçları olabilir. İşte bu dezavantajlar net hesaplanamaz. Burada kumaşın boyu, maliyet ve memnuniyeti dengeleyen bir fonksiyon ile hesaplanır. Hesaplanan kumaş boyları baz alınarak, problem çözülmüştür. Şekil 4.1'de bir örnek çözüm yapılmış ve kazanım resmedilmiştir (Kim ve diğer., 2001).



Şekil 4.1 Sol tarafta kutulama ile çözüm, sağda ise bulanık kutulama ile çözüm

## 4.2 Bulanık Yeterlilik Matrisi ile En Uygun Görev Dağılımı Problemi

Görevlere obje, görevlerin atamalarının yapıldığı kişi veya gruplara kutu denirse problem kutulama problemine evrilir. Burada obje ile kutu arasındaki ilişki $c_{ij}$ , $i = 1, \dots, m$ , $j = 1, \dots, n$ olacak şekilde bir matris içerisinde ifade edilmiştir.

$$x_{ij} = \begin{cases} 0\,; & i\ objesi,\ \ j\ kutusuna\ konmamıştır. \\ 1\,; & i\ objesi,\ \ \ j\ kutusuna\ konmuştur. \end{cases} \tag{4.1}$$

Bu notasyonlardan sonra atamanın kalitesi;

$$\gamma(x) = \min x_{ij} c_{ij} \tag{4.2}$$

şeklinde hesaplanır ve amaç bu değeri maksimum yapmaktır. Bu amaca ulaşmak için 2 kademeli bir algoritma oluşturulmuştur. Algoritmanın 1 kısmı FFD kullanmaktadır. İlk kısım bittiğinde oluşan atamalar ikinci algoritmaya parametre olarak iletilir. Algoritmanın ikinci kısmı, atanmış elemanlardan kalite değeri en küçük olanı bulur. Bulunan en küçük ile boyut kıstaslarına dikkat ederek diğerlerini karşılaştırır. Yer değiştirildiğinde, toplam kaliteyi yükselten bir başka eleman bulursa o eleman ile yer değişimi yapar ve sonraki en küçük kaliteye sahip eleman ile devam eder. Eğer değişim yapacak eleman bulunmazsa algoritma sonlanır (Nasibov ve diğer., 2004).

### *4.2.1 Algoritma 1-0 (Başlangıç Çözümün Bulunması Aşaması)*

$x_{ij}\ i = 1\ , \dots , m + 1\ ; j = 1, \dots , n$ matrisi oluşturulur.

$c_{ij}, i = 1\ , \dots , m; j = 1\ , \dots , n$; yeterlilik matrisi.

$a_j, j = 1\ , \dots , n$; objelerin boyutu.

$b_i, i = 1\ , \dots , m$; kabul edilir maximum yük.

$s_i = b_i, i = 1, \dots , m$; kutularda kalan boş alan

*DegMin* = 1, tüm yerleştirmenin yeterlilik derecesi

$i = 0$; ilk değerleri ata.

*Objeleri boyutlarına göre azalan sırada sırala

*Kutuların maximum yük kapasiteleri dikkate alınarak kutularda ki boş alanlara göre azalan şekilde sırala. (***)

**for** $j = 1\ to\ n$ **begin**

  **for** $i = 1\ to\ m$ **begin**

    **if** ( $a_j \leq s_i$ ) **begin**

      $x_{ij} = 1$;  // i. objeyi j. kutuya koy.

      $s_i = s_i - a_j$; //kutunun kalan kısmını güncelle.

      //yeterlilik derecesini güncelle

      DegMin = min$\{\ DegMin\ , c_{ij}\ \}$;

      **break**;

    **endif**

  **endfor**

**endfor**

$x_{ij}, i = 1\ , \dots , m + 1; j = 1, \dots , n$ //atamaların son halini içeren matris.

### *4.2.2 Algoritma 1-1 (Çözümün İyileştirilmesi Aşaması)*

$x_{ij},\ i = 1, \dots , m + 1; j = 1, \dots , n$; bu matris birinci algoritmadan elde edilir.

$c_{ij}, i = 1\ , \dots , m; j = 1\ , \dots , n$; yeterlilik matrisi.

$a_j, j = 1\ , \dots , n$; objelerin boyutu.

$b_i, i = 1\ , \dots , m$; kabul edilir maximum yük.

$s_i = b_i, i = 1, \dots, m$; kutularda kalan boş alan

DegMin = 1, tüm yerleştirmenin yeterlilik derecesi

$devam = true$; döngünün devamını kontrol için.

**while** ( $devam == true$ ) **begin**

/* Yerleştirilmiş elemanlar arasında ki en küçük dereceyi bul */

DegMin = $c_{i_0 j_0} = \min\{c_{ij} | \, x_{ij} = 1, i = 1, \dots, m; j = 1, \dots, n\}$;

$devam = false$;

**for** $i = i_0 \; to \; n$ **begin**

**if** ( $x_{ij}$ ! = 1) **continue**;

**if** ( $a_{ij} + s_i < a_{i_0 j_0}$ ) **continue**.

**if** ( $a_{i_0 j_0} + s_{i_0} < a_{ij}$ ) **continue**.

**if** ($\min\{c_{i_0 j}, c_{i j_0}\} \leq c_{i_0 j_0}$ ) **continue**.

// değiştirme gerçekleşir.

$x_{i j_0} = 1$;

$x_{i_0 j} = 1$;

$x_{i_0 j_0} = 0$;

$x_{ij}$ = 0;

$devam = true$;

**break**;

**endfor**

**endwhile**

// güncelleştirmeleri içeren matris

$x_{ij}$, i = 1 , … , $m + 1; j = 1, \dots, n$.

Daha sonraki yıllarda yine Nasibov ve Kınay'ın beraber yayınladığı bir makalede 1. algoritmada (***) ile işaretli yerde bulunan sıralama algoritması dinamik bir şekilde her bir objede ayrı hesaplanıp sıralanmıştır. Sıralama döngü içine alınıp o anki iş dikkate alınarak sıralama yapılmış ve yapılan test sonuçlarında algoritmanın ikinci aşamasının daha az iyileştirme yaptığı yani birinci aşamanın daha iyi sonuçları aşama 2'ye ilettiği gözlenmiştir (Nasibov ve Kınay, 2006).

## 4.3 Bulanık İlişki Matrisleri İle Kutulama Problemi

X = $\{x_1, x_2, \dots, x_n\}$ objelerden oluşan bir küme olsun ve S = $\{S_1, S_2, \dots, S_m\}$ kümesi ise kutuları ifade etsin ve $S_{m+1}$ kutusu ise asıl kutular olan $S_i, i = 1, \dots, m$ kutularına yerleşmeyecek elemanları barındıracak olan kutudur. Objeler veya kutular arasındaki bulanık ilişki ise şu şekilde ifade edilmiştir;

$$R_1 = \begin{bmatrix} r_{11} & \cdots & r_{1n} \\ \vdots & \ddots & \vdots \\ r_{\mathrm{n1}} & \cdots & r_{nn} \end{bmatrix}, i, j = 1, \dots, n; r_{ij} = R_1(x_i, x_j) \tag{4.3}$$

matrisi simetrik ve yansıma özelliklerine sahiptir ve $x_i \; ve \; x_j$ objeleri arasındaki bağı ifade eder.

$$R_2 = \begin{bmatrix} r_{11} & \cdots & r_{1n} \\ \vdots & \ddots & \vdots \\ r_{\mathrm{n1}} & \cdots & r_{nn} \end{bmatrix}, i, j = 1, \dots, n; \; r_{ij} = R_2(x_i, x_j) \tag{4.4}$$

matrisi simetrik ve yansıma özelliklerine sahip olup objeler arasındaki uyumluluğu ifade eder.

$$R_3 = \begin{bmatrix} r_{11} & \cdots & r_{1m} \\ \vdots & \ddots & \vdots \\ r_{\mathrm{n1}} & \cdots & r_{nm} \end{bmatrix}, i = 1, \dots, n, j = 1, \dots, m; \;\; r_{ij} = R_3(x_i, S_j) \tag{4.5}$$

matrisi $x_i$ objesi ise $S_j$ kutusu arasındaki bağı ifade eder.

$$R_4 = \begin{bmatrix} r_{11} & \cdots & r_{1m} \\ \vdots & \ddots & \vdots \\ r_{\mathrm{n1}} & \cdots & r_{nm} \end{bmatrix}, i = 1, \dots, n, j = 1, \dots, m; \;\; r_{ij} = R_4(x_i, S_j) \tag{4.6}$$

matrisi $x_i$ objesi ise $S_j$ kutusu arasındaki uyumluluğu ifade etmektedir.

Belirtilen bütün matrislerdeki değerler [0,1] aralığında değer almaktadır. Uyum derecesini ifade eden matrislerde 0 değeri bu objelerin beraber olamayacağını ifade ederken, 1 değeri ise objelerin tamamen serbest olduğunu ifade etmektedir. Bağ matrislerinde ise 1 değeri objelerin birbirleri ile bağımlı yani beraber olmaları

gerektiğini ifade ederken 0 değeri ise objelerin bağımsız yani beraber olup olmama konusunda serbest olduklarını ifade etmektedir. $R_i, i = 1, \dots, 4$ matrislerinin değerleri karar verici tarafından önceden girilmelidir.

Dikkat edilirse $S_{m+1}$ kutusu herhangi bir kısıt içermemektedir. Ayrıca X kümesinin elemanı olan objeler $Q_1, Q_2, \dots, Q_k$ gibi gruplarda toplanabilirler. Bu grupların arasındaki ilişki matrisleri tekrardan oluşturulur.

Kutular, önceden belirlenmiş bazı koşulları sağlayan objeler ile doldurulmalıdır. Bu şekilde, kalitenin derecesi yani tutarlılığın derecesi maksimize edilmeye çalışılırken $S_{m+1}$ kutusunun içeriği minimize edilmelidir.

Bazı notasyonlar şu şekilde tanımlanıştır;

$$K_1(S_j) = 1 - \max\{R_1(x^1, x^2) |\, x^1 \in S_j\,, x^2 \notin S_J\}\,, \quad j = 1, \dots, m. \quad (4.7)$$

$$K_2(S_j) = \min\{R_2(x^1, x^2) | x^1 \in S_J, x^2 \in S_j\}\,, \quad j = 1, \dots, m. \quad (4.8)$$

$$K_3(S_j) = 1 - \max\{R_3(x, S_j) | x \notin S_J\}\,, \quad j = 1, \dots, m. \quad (4.9)$$

$$K_4(S_j) = \min\{R_4(x, S_j) | x \in S_J\}\,, \quad j = 1, \dots, m. \quad (4.10)$$

Burada $K_1(S_j)$, $S_j$ kutusu içindeki objeleri dışındaki objelerden ayırmanın tutarlılık derecesini vermektedir. $K_2(S_j)$, $S_j$ kutusunun içindeki objelerin birbiri ile uyumluluk derecesini ifade eder. $K_3(S_j)$, $S_J$ kutusunun dışında kalan objeleri o kutudan ayırmanın ne kadar tutarlı olduğunu gösterir. $K_4(S_j)$, $S_j$ kutusu içindeki objelerin, kutu ile olan uyumluluk derecesini ifade etmektedir.

Bütün kutulara yerleşmiş objeler üzerinden genel kalite ise;

$$A = \min_{j = 1 \dots m} \min\{\, K_1(S_j)\,, K_2(S_j)\,, K_3(S_j)\,, K_4(S_j)\,\} \quad (4.11)$$

şeklinde hesaplanmıştır. Kısıtları ise;

$$F_i\left(x\middle|x \in Q_i \cap S_j\right) \subsetneq B_{ij}, i = 1, \dots, k; j = 1, \dots, m. \quad (4.12)$$

$F_i$ fonksiyonu $x \in Q_i \cap S_j$ üzerinde herhangi bir doğrusal fonksiyondur. $B_{ij}$ ise konveks bir bulanık kümedir. $\subsetneq$ bağıntısı ise farklı biçimlerde tanımlanabilen bulanık küme karşılaştırma işlemlerinden herhangi biri olabilir.

Bulanık kutulama problemi şu şekilde formüle edilebilir.

$$A \to max, \quad (4.13)$$

$$\sum_{x \in S_{m+1}} F(x) \to min, \quad (4.14)$$

$$F_i\left(x\middle|x \in Q_i \cap S_j\right) \subsetneq B_{ij}, i = 1, \dots, k; j = 1, \dots, m \quad (4.15)$$

(4.13)-(4.15) kısıtları altında X kümesinin elemanlarını $S_j, j = 1, \dots, m$ kutularına dağılımı aranmaktadır. A kalite derecesi (4.11)' de tanımlanmıştır ve $F_i(x), x = 1, \dots, k,$ ise doğrusal fonksiyonlardır. Kutulamanın maksimum derecesi hakkında, öncelikli tahminde bulunulabilmektedir. (4.13)-(4.15) ifadelerinde tanımlanan problemi çözmek için en pratik yaklaşımlardan olan ayrıştırma yaklaşımı kullanılmaktadır. (4.13)-(4.15) arasını şu şekildedir;

$$A \geq g, \quad (4.16)$$

$$\sum_{x \in S_{m+1}} F(x) \to min, \quad (4.17)$$

$$F_i\left(x\middle|x \in Q_i \cap S_j\right) \subsetneq_g B_{ij}, i = 1, \dots, k; j = 1, \dots, m \quad (4.18)$$

$$g \in (0,1], \quad (4.19)$$

Burada $\subsetneq_g$ ifadesi 'g' derecesinden kısıtların sağlanmasıdır. (4.16)-(4.19) probleminin 'g' dereceden çözümü, (4.16)-(4.19) probleminin genel çözümünün g-seviye kümesidir. Dolayısıyla, ayrıştırma (dekompozisyon) prensibine göre, (4.16)-(4.19) probleminin genel çözümü g-seviye kümelerinin üyelik dereceleri bazında birleşimidir.

Şimdi (4.16)-(4.19)'ün çözümü üzerinde çalışılacaktır. Aşağıdaki Lemma doğrudur. Verilen lemma ve teoremlerin ispatı için bakınız: (Nasibov, 2004).

*Lemma 1.* Eğer bir $g \in [0,1]$ değeri için herhangi $x_i \in X$ $obje$ $ve$ $S_j \in S$ kutusu için

$$\max\{1 - R_3(x_i, S_j), R_4(x_i, S_j)\} < g \tag{4.20}$$

sağlanıyorsa, o 'g' değeri için problem (4.13)-(4.15)'ün bir çözümü yoktur.

Teorem 1.

$$g > \min_{\substack{i=1\dots n \\ j=1\dots m}} \max\{1 - R_3(x_i, S_j), R_4(x_i, S_j)\} \tag{4.21}$$

gibi hesaplanan 'g' değeri için çözüm yoktur.

*Teorem 2.* $R_1'$ matrisi $R_1$ matrisinin kapalı geçişli hali olsun. Bu durumda

$$g > \min_{i,j=\ 1\dots n} \max\{1 - R_1'(x_i, x_j), R_2(x_i, x_j)\} \tag{4.22}$$

ifadesini sağlayan 'g' değerleri için çözüm yoktur.

Teorem 1 ve 2'nin kullanılması ile verilecek algoritmanın hesaplamaları hızlandırılabilir.

Problem (4.13)-(4.15) için kabul edilebilir sonucu bulmak için sezgisel bir algoritmadan bahsedilecektir. 'First Fit Decreasing (FFD)' algoritması temel alınarak tasarlanmış bir algoritmadır. Bu yüzden objeler önceden hacim ve kısıtlara göre azalacak şekilde sıralanmıştır. Hesaplamayı hızlandırmak için önce $R_1'$ matrisindeki bağımlılık derecelerine göre g-grupları elde edilecektir.

**Algoritma 2**. Objelerin kutulara dağılımı.

| *$R_1'$ kapalı geçişli matrisinin $R_1$'den elde edilir.

*Teoremlerden faydalanarak matrislerden elde edilebilecek en yüksek kalite derecesi hesaplanır.

$g_1 = \min\limits_{i,j=1\ldots n} \max\{1 - R_1'(x_i, x_j), R_2(x_i, x_j)\}$ hesaplanır.

$g_2 = \min\limits_{\substack{i=1\ldots n \\ j=1\ldots m}} \max\{1 - R_3'(x_i, S_j), R_4(x_i, S_j)\}$ hesaplanır.

$g = \min\{g_1, g_2\}$; elde edilebilecek en yüksek kalitedir.

**tekrar**:

*****G-seviyeların oluşturulması; Gruplar $y_1, y_2, \ldots, y_l, l \leq k$ şeklinde ifade edilir ve $x^1, x^2 \in y_i \; ve \; x^3 \notin y_i$ olacak şekilde $R_1'(x^1, x^2) \geq g \; ve \; R_1'(x^1, x^3) < g$ şartını sağlayacak şekilde gruplar oluşturulur.

***** Gruplar oluşturulduktan sonra bu grupların aralarındaki bulanık bağlantıyı ifade eden $R_2', R_3' \; ve \; R_4'$ matrisleri oluşturulur.

$R_2'(y_i, y_j) = \min\{R_2(x^1, x^2) | x^1 \in y_i, x^2 \in y_j\}$;

$R_3'(y_i, S_j) = \min\{R_3(x^1, x^2) | x^1 \in y_i, x^2 \in S_j\}$;

$R_4'(y_i, S_j) = \min\{R_4(x^1, x^2) | x^1 \in y_i, x^2 \in S_j\}$;

//$R_3'$ ilişkisine dayalı ön yerleştirme yapılır.

**for** $i = 1 \; to \; l$ **begin**

  **for** $j = 1 \; to \; m$ **begin**

    **if** ( $R_3'(y_i, S_j) \geq 1 - g$ ) **begin**

      **if** ( $R_4'(y_i, S_j) < g$ ) **or**

        ( $\min\{R_2(y_i, y) | y \in S_j\} < g$ ) **or**

        ( $y_i \; S_j \; içine \; girmezse$ ) **begin**

        *problemin çözümü yok, 'g' değerini düşür

        ***tekrar***'a git.

      **endif**

      $*y_i \; objesini \; S_j \; içine \; yerleştir.$

    **endif**

  **endfor**

**endfor**

**for** $i = 1 \; to \; n$ **begin**

  **if** ($y_i \; yerleşmişse$) **continue;**

**for** $j = 1$ *to* $m$ **begin**

**if** $\left( R_4'\left(y_i, S_j\right) \geq g \right)$ **and**

$(\min\{R_2(y_i, y) | y \in S_j\} \geq g)$ **and**

( $y_i, S_j$ *kutusuyla* (2.3) *şartını sağlıyorsa* ) **begin**

* $y_i$ objesi $S_j$ içine yerleştir.

**break**;

**endif**

**endfor**

**if** ( $y_i$ *yerleşmişse* ) **continue**;

* $y_i, S_{m+1}$ kutusuna koy.

**endfor**.

$S_{m+1}$ kutusu için bir kısıt olmadığı için Algoritma 3 her zaman kabul edilebilir bir sonuç bulur.

Teorem 3.

$$g_1 = 1 - \max\{R_1(x^1, x^2) | x^1 \in X\,, x^2 \in X\,\}, \tag{4.23}$$

$$g_2 = \min\{R_2(x^1, x^2) | x^1 \in X\,, x^2 \in X\,\}, \tag{4.24}$$

$$g_3 = 1 - \max\{R_3(x^1, s) | x^1 \in X\,, s \in S\,\}, \tag{4.25}$$

$$g_4 = \min\{R_4(x^1, s) | x^1 \in X\,, s \in S\,\}, \tag{4.26}$$

Herhangi bir $g < g^0 = min\{g_1, g_2, g_3, g_4\}$ için problemin bir çözümü yoktur. (Nasibov, 2004)

# BÖLÜM BEŞ
# BULANIK KISITLAR İLE TAKIM OLUŞTURMA

Bulanık ilişkilerle kutulama probleminin bir alt problemi olarak insanların takımlara veya öğrencilerin projelere yerleştirilmesi problemini örnek gösterebiliriz. Örneğin bir sınıftaki öğrencilerden matematik yarışması için takımlar oluşturulabilir ve bu takımların dağılımının kalitesinin hesaplanması istenebilir. Burada takımlar içinde bir tane bile öğrencinin matematik konusunda yetenekli olması o takımı başarılı bir takım yapmaya yetecektir. Hatta bu başarılı takım daha sonra bir olimpiyata katıldığında sağladığı başarı okulun başarısı olarak geçecek ve okul 5 sorudan 5'inide bilerek tam not aldı şeklinde anılacaktır. Aynı şekilde kimya veya benzeri dersler içinde takımlar oluşturulabilir. Fakat okulun genel başarısı sorgulandığında tüm takımlar arasında en kötü puanı alan takım üzerinden konuşmak veya bunların ortalamasını almak daha anlamlı olacaktır.

O halde; bazı durumlarda takımın içindeki bireylerin hepsinin en iyi olması geçerli bir kontrol olmayabilmektedir. Bu yeni problemde dağılımın kalite değeri hesabı, Nasibov 'un bulanık kutulama modelinde hesaplanan şeklinde farklı olacaktır. Bu yeni ele alacağımız problemin, kalite hesaplaması için aşağıda yeni bir yaklaşım önerilmiştir.

## 5.1 Problemin Açıklaması

$X = \{x_1, x_2, \dots, x_n\}$ kişilerinden oluşan bir küme ve $S = \{S_1, S_2, \dots, S_m\}$ ise takımları ifade etmektedir. Kişiler ile takımlar arasındaki bağ şu şekilde tanımlanmıştır;

$$R_3 = \begin{bmatrix} r_{11} & \cdots & r_{1m} \\ \vdots & \ddots & \vdots \\ r_{\mathrm{n}1} & \cdots & r_{nm} \end{bmatrix}, i = 1, \dots, n\ , j = 1, \dots, m \ ve \ r_{ij} = R_3(x_i, S_j) \qquad (5.1)$$

matrisi $x_i$ kişisiyle $S_j$ takımı arasındaki bağı ifade eder.

$$R_4 = \begin{bmatrix} r_{11} & \cdots & r_{1m} \\ \vdots & \ddots & \vdots \\ r_{\mathrm{n}1} & \cdots & r_{nm} \end{bmatrix}, i = 1, \dots, n\ , j = 1, \dots, m\ ve\ r_{ij} = R_4(x_i, S_j) \quad (5.2)$$

matrisi $x_i$ kişisiyle $S_j$ takımı arasındaki uyumluluğu ifade etmektedir.

Görüldüğü üzere, bu modelimizde Nasibov'un genel modelindeki 1. ve 2. matrisler kullanılmamaktadır. Numaralandırmayı bozmamak adına matrislerin isimleri $R_3$ ve $R_4$ olarak bırakılmıştır.

Belirtilen matrislerdeki değerler [0,1] arasında değer almaktadır. $R_3$ matrisinde yer alan 1 değeri kişinin o takıma mutlak girmesi gerektiğini ifade ederken 0 değeri ise kişinin o takıma girip girmemekte serbest olduğunu ifade eder. $R_4$ matrisi ise uyumluluk matrisidir. Eğer değer 0 ise o kişi bu takım ile kesinlikle uyumlu değildir. 1 ise o takım ile uyumludur ve girip girmemekte serbesttir. Karar verici, bu matrisleri önceden doldurmalıdır.

Takımlar, kişiler ile doldurulacak ve dağılımın kalitesi maksimize edilmeye çalışılacaktır. Bu kalite hesabında kullanılan notasyonlar şu şekildedir.

$$\mu_{ij} = \begin{cases} 1 - R_3(x_i, S_j), & x_i \notin S_j \\ R_4(x_i, S_j), & x_i \in S_j \end{cases} \quad (5.3)$$

Fonksiyonda; $\mu_{ij}$, $x_i$ kişisinin $S_j$ takımına göre kalite derecesini göstermektedir. Daha sonra yerleştirme durumuna göre kalite hesabı şu şekillerde hesaplanabilir.

$$\boldsymbol{A_1} = \min_{j=1\dots m} \min_{i=1\dots n} \mu_{ij} \quad (5.4)$$

en düşük kaliteye sahip takımın, en düşük dereceden kişisi temel alınarak yapılan kalite hesabıdır. Burada kalitenin yükselmesi, tüm kişi ve takımların kalitesinin yüksek olduğunu garanti eder. Nasibov makalesinde değerlendirmesini bu yaklaşımı kullanarak yapmıştır.

$$\boldsymbol{A_2} = \min_{j=1\ldots m} \max_{i=1\ldots n} \mu_{ij} \tag{5.5}$$

bir takımda bir tane bile derecesi yüksek kişi bulunursa o; takımın derecesini yükseltir fakat genel değerlendirme en düşük dereceden takıma göre yapılır. Tek branş takımları oluşturulacakken kullanılması yerindedir.

$$\boldsymbol{A_3} = \max_{j=1\ldots m} \min_{i=1\ldots n} \mu_{ij} \tag{5.6}$$

takımın içindeki öğrencilerin en düşük dereceli olanı baz alınır. Ancak bu takımlar arasında en başarılı olan takım tüm kaliteyi belirler.

$$\boldsymbol{A_4} = \max_{j=1\ldots m} \max_{i=1\ldots n} \mu_{ij} \tag{5.7}$$

en iyimser yaklaşımdır. Bir tane derecesi yüksek öğrenci tüm takımı, bir tane derecesi yüksek takım tüm dağılımın kalitesini yükseltir.

Bir okul, hem okul dersleri hem de spor dalları üzerinden yarışmaya katılacak olsun. Dersler fen bilimleri, fizik, kimya, biyoloji; sporlar da basketbol ve futbol olsun. Katılacakları yarışmaya gitmek için en iyi takımlarını hazırlayacaklar bunun için önce her dalda birden çok takım oluşturacaklardır. Önce ders veya spordan birden çok takım oluşturulur. Bu takımların arasından en iyi seçilir. Dersler üzerine kurulan takımlar için; takımların içinden iyi olan bir öğrenci bile takımını başarıya götüreceği için $A_4$ formülü ile dağılım derecesi hesaplanması tercih edilebilir. Diyelim ki, okulun, bir spor müsabakasına takım göndermek için ön eleme yapması gerekmektedir. Bu takımların başarılı olması için takımlara ait tüm üyelere görev düşmektedir ve kalite, en kötü dereceli öğrenci baz alınarak hesaplanması uygundur. Takımlardan bir tanesinin başarılı olması ön eleme için yeterlidir. Burada $A_3$ formülünün kullanılması uygundur. Okulun derslerdeki genel başarısı $A_2$ formülü ile hesaplanır, takım içinde bulunan bir öğrencinin başarısı o takımı başarıya ulaştırmaya yetecekken; tüm branşlar arasında en düşük dereceye sahip olan takım okulun derecesini yansıtmış olur. Yarışmada okulun spor dallarında ki genel başarısına bakılacak olursa $A_1$ formülü istenilene uygun olacaktır. Takımlardaki herkesin başarılı olması aranırken tüm dallardaki takımlardan da en düşük dereceye sahip olan baz alınmaktadır.

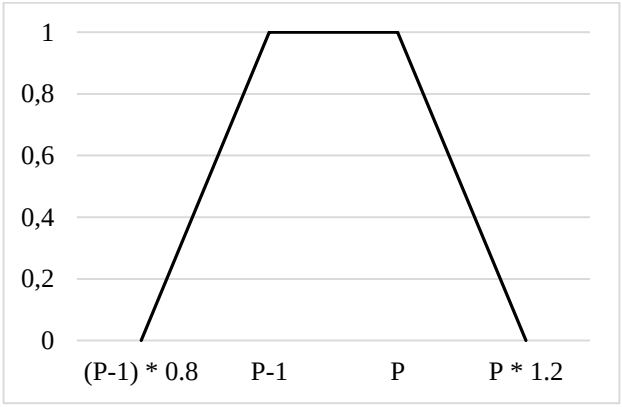
Görüldüğü gibi sunulan $A_i, i = 1 \dots 4$, formüllerinin hepsi de farklı bir durumda ortaya çıkan arayışa cevap vermektedir. Takımların kapasitesinin eşit olması gerekmez. Dolayısı ile takımların kapasiteleri bulanık bir sayı ile ifade edilmiştir. Bununda mümkün olan en homojen şekilde dağıtılması hedeflenmiştir. Homojen dağılımın kalitesi 'N' olarak ifade edilecektir. Kalite derecesi Şekil 5.1'de bahsedilen bulanık sayının üyelik derecesidir.

Kısıtları ise; $A \equiv A_i,\ i = 1 \dots 4$ olmak üzere seçilen bir 'i' değeri için;

$$A \to max \tag{5.8}$$

$$N = \min_{j=1\dots m} \mu_{\bar{N}}(\sum_{i=1\dots n} Z_{ij}) \ ,\ Z_{ij} = \begin{cases} 0, x_i \notin S_j \\ 1, x_i \in S_j \end{cases} \tag{5.9}$$

şeklindedir. Z matrisi hangi kişinin hangi kümeye yerleştiği verisini tutar. (4.13)-(4.15) probleminde (4.14) kısıtı ekstra kutu olmaması nedeni ile kaldırılmıştır. Burada da mümkün olduğunca takım üyeleri sayısı birbirine yakın olması amaçlanmıştır. Ancak daha yüksek bir 'A' değeri elde edebilmek için bu bulanık bir ilişki olarak tanımlanmıştır.




Şekil 5.1 Takımların kapasitesini belirten bulanık sayı

$K = \left\lceil \frac{n}{m} \right\rceil$ olacak şekilde bir değer alınır. Bu değer ile takımların eleman sayıları bulanık sayı şeklinde gösterilir.

$\bar{N} = [a, b, c, d]$ olacak şekilde yamuk biçimli bulanık sayı şeklinde ifade edilir.

$N_j = \overline{N}(S_j)$ olacak şekilde $S_j$ takımının eleman sayısının $\overline{N}$ bulanık sayısına üyelik değerini ifade eder.

$N = \min_{j=1...m} N_j$ ise takımların kapasitelerinin kullanımına göre kalite derecesini verir ve genel kalite derecesi

$$Q = \min\{N, A\} \tag{5.10}$$

şeklinde tanımlanmıştır.

## 5.2 Hesaplama Deneyleri İçin R-Matrislerinin Oluşturulması

Algoritmaları test etmek için gerekli olan problem kümeleri, daha önce üzerinde çalışılmadığından yoktur. Bu kümelerin farklı parametreler altında oluşturulması için bir algoritma geliştirilmiştir. Algoritma şu şekildedir;

ES: kişi sayısı, girdi;
TS: takım sayısı, girdi;
$f_3$: $R_3$ için serbestlik faktörü (float) [0,1] 0: serbest değil, 1: serbest;
$f_4$: $R_4$ için serbestlik faktörü (float) [0,1] 0: serbest değil, 1: serbest;
MD: rastgele üretilmiş çözüm kümesi
RA(): [0,1] arası rastgele değer üreten fonksiyon
$f_a$: bulanık sayı genişlik faktörü (float) [0,1], girdi;
TQ: hedef kalite, (float) [0,1] , girdi, normal: 0.8;
/* Model dağılımı oluştur */
fuzzyDif = $\left\lceil \frac{ES}{TS} \right\rceil * f_a$;
maxItem = $\left\lceil \frac{ES}{TS} \right\rceil$ + fuzzyDif;
minItem = $\left\lfloor \frac{ES}{TS} \right\rfloor$ - fuzzyDif;
**if** ( fuzzyDif > 0 )
    $\overline{N}$ = [ $minItem * 0.5$ , $minItem$ , $maxItem$, $maxItem * 1.3$ ];
**else**

$\bar{N} = [\ minItem, minItem, maxItem, maxItem\ ]$;

**endif**

MD = *$\bar{N}$ sayısını dikkate alarak rastgele dağılım oluştur;

/* Model dağılımı baz alarak R matrislerini oluştur */

**for** $i = 1\ to\ ES$ **begin**

**for** $j = 1\ to\ TS$ **begin**

$R_3(i,j) = (1 - TQ) * RA()$;

$R_4(i,j) = TQ * RA()$;

**if** ( $MD'ye\ göre\ i\ \in j$ ) **continue**;

**if** ( $RA \geq f_3$ ) $R_3(i,j) += RA() * TQ$;

**if** ( $RA \geq f_4$) $R_4(i,j) += RA() * (1 - TQ)$;

**endfor**

**endfor**.

Algoritma çalıştırıldığında $R_3$ ve $R_4$ matrisleri parametrelere göre oluşturulmaktadır. Parametreler ise;

- TQ, model dağılım için hedeflenen kalite değeridir.
- ES ve TS matrislerin boyutları için gereklidir.
- Faktör değerleri olan $f_3$ ve $f_4$ ise oluşturulan $R_3$ ve $R_4$ matrislerinin ne kadar serbest olacağını belirler. 0 değeri verildiğinde üretilen matrisler ile dağılım kalitesi, sadece model dağılım için TQ değerini verecektir. 0 değerinden 1'e yaklaştıkça model dağılımına bağımsız değerler üretilir. Bir eleman, model dağılımda girmediği bir takıma koyduğunuzda kalite düşmeyebilir.
- $f_a$, takım kapasiteleri için bulanık sayı oluşturulurken kullanılır. Eğer 0 ise takım kapasiteleri homojen ayarlanır, yani yamuksal bulanık sayının tavan genişliği maksimum 2'dir. Değer yükseldikçe, takımların eleman sayılarındaki farklılıklar kaliteyi daha az etkiler.

Algoritma; önce verilen eleman sayısı ve takım sayısını dikkate alarak rastgele bir dağılım oluşturur. Bu oluşan dağılıma 'model dağılım' adı verilmiştir. Algoritma bu dağılıma göre kullanılacak $R_3$ ve $R_4$ matrislerini oluşturacaktır. Verilen $f_3$ ve $f_4$ faktör değerleri bulanık bir ilişki ile oluşturulacak matrislerin model dağılım ile ilişkisini

daha kesin veya daha esnek tutacaktır. Örneğin; model dağılımda ‘a’ objesi 1. kutuda olsun. Algoritma, $f_3$ ve $f_4$’ü bir şans faktörü olarak alır, 0 ile 1 arasında bir sayı üretir eğer sayı bu faktörlerden büyük ise matrislere değerler yazılır. Eğer koşul sağlandı ise (örneğin faktörler 0 değerine sahip) $R_3$ bağ matrisi, model dağılımdaki takımlarda olan kişileri birbirine bağlayan değerler ile doldurulur, $R_4$ uyum matrisinde ise takımlar ile o takıma ait olmayan kişilerin dereceleri uyumsuz olarak işaretlenir. Faktörlere 0’dan büyük değer girildikçe $R_3$ matrisine girildiğinde bağ oluşturan değerler atlanır, $R_4$ uyum matrisinde ise uyumsuz işaretlenecek elemanlar uyumsuz işaretlenmez. Her iki durumda da model dağılımın kalitesi test edilecek algoritmaların bulduğu kaliteleri ölçmek için baz alınmaktadır. Model dağılım tablolarda, önerilen algoritmaların sonuçları ile karşılaştırılmaya konulmuştur. Algoritmalar model dağılım kalitesinde dağılım yaptığında başarılı olduğu kabul edilebilir.

## 5.3 Problemin Çözümü İçin İlk Öneriler

İlk olarak problemden kaliteli bir sonuç çıkartmak için *N* ve *A* olmak üzere iki farklı değeri iyileştirmek gerekmektedir. *N* takımların eleman sayılarını istenilen aralıkta tutmak anlamına gelirken *A* ise takımlara dağıtılan elemanların *R* ilişki matrislerinde ki ilişkilere en uygun şekilde dağıtılması anlamına gelmektedir. İlk yaklaşım olarak bu iki değerden N değerinin yüksek olması için takım kapasiteleri, homojen dağılım sağlayacak şekilde verildi. Bu yöntem ile çalışan 5 algoritma üretilip, bu algoritmaların performansları sunulacaktır.

### *5.3.1 Algoritma 1: Temel Algoritmanın Yeniden Yorumlanması*

Temel algoritma içinde bulunan ve bu çalışmada kullanılmayan diğer matrisler ile yapılan gruplandırma işlemleri gibi işlemler çıkarıldı. Kutuların hepsinin eleman sayısının benzer olması kaliteyi etkilediği için algoritmalarda hep eleman sayılarının benzer olmasını garantileyen bir yol izlendi. K değişkeni bir kutunun alabileceği maksimum eleman sayısını belirler. Bir kutunun eleman sayısı K’yı geçecek olursa daha eleman almaması için kontrol eklendi. Böylelikle (5.9) kısıtı her zaman sağlanmış oldu. Algoritma 1, olabilecek en yüksek kalite değerinden başlayıp elemanları o

kaliteye uygun yerleştirilebilir mi diye kontrol eder. Eğer kontrol başarısız olursa; hedef kalite değerini düşürüp, yeni kalite değeri için tekrar dener.

$g = g_{max}$;
$K = \left\lceil \frac{n}{m} \right\rceil$;
$S_j,\ j = 1 \ldots m$ *takımları boşalt.*
tekrar:
**for** $i = 1$ *to* $n$ **begin**
    **for** $j = 1$ *to* $n$ **begin**
        **if** $\left( S_j' nin\ eleman\ sayısı \geq K \right)$ **continue**;
        **if** $\left( R_3(x_i, S_j) \geq 1 - g \right)$ **begin**
            **if** $\left( R_4(x_i, S_j) < \mathrm{g} \right)$ **begin**
                *g değerini azalt.
                ****tekrar***'a git.
            **endif**
            *$x_i$ kişisini $S_j$ kutusuna yerleştir.
        **endif**
    **endfor**
**endfor**
**for** $i = 1$ *to* $n$ **begin**
    **for** $j = 1$ *to* $m$ **begin**
        **if** $\left( S_j' nin\ eleman\ sayısı \geq K \right)$ **continue**;
        **if** $\left( R_4(x_i, S_j) > g \right)$ **begin**
            *g değerini azalt.
            ****tekrar***'a git.
        **endif**
    **endfor**
**endfor**.

### *5.3.2 Algoritma 2: Bağımlılık Matrisine Göre Sıralı Sezgisel Yaklaşım*

Algoritma FFD algoritmasından esinlenerek oluşturuldu. Sıralama, klasik kutulama probleminde objelerin ebatlarına göre yapılmaktaydı. Ancak çözmek istenilen modelde kişiler 1 birim yer kaplamaktadır. Yerleştirilecek kişi için takımlar, $R_3$ matrisindeki değere göre sıralanır, kişi sıraya göre ilk müsait takıma yerleştirilir. Uyum matrisi daha serbest olup bağımlılık matrisi detaylandırılmış olan durumlar hızlı ve yüksek kaliteli dağılımlar yapabilmektedir.

$K = \left\lceil \frac{n}{m} \right\rceil$;
**for** $i = 1$ *to* $n$ **begin**
    Sıralı =Kutuları $1 - R_3(i, kutu_{index})$ değeri için azalan sırala;
    **for** $j = 1$ *to* $m$ **begin**
        **if** $(Sıralı(j)\ eleman\ sayısı \geq K)$ **continue**;
        *$i$ elemanı $Sıralı(j)$ kutusuna yerleştir;
        **break**;
    **endfor**
**endfor**.

### *5.3.3 Algoritma 3: Uyumluluk Matrisine Göre Sıralı Sezgisel Yaklaşım*

Algoritma FF algoritması gibi tasarlanmıştır. Algoritma 2'den farkı $R_4$ algoritmasına göre kutular sıralanmış ve objeler boş yer varsa yerleştirilmiştir. Uyum matrisinin detaylandırıldığı durumlarda hızlı çözümler sunabilmektedir.

$K = \left\lceil \frac{n}{m} \right\rceil$;
**for** $i = 1$ *to* $n$ **begin**
    *Sıralı = Kutuları $R_4(i, kutu_{index})$ değeri için artan sırala;
    **for** $j = 1$ *to* $m$ **begin**
        **if** $(Sıralı(j)\ eleman\ sayısı \geq K)$ **continue**;
        $i$ elemanı $Sıralı(j)$ kutusuna yerleştir;

**break**;
**endfor**
**endfor**.

### *5.3.4 Algoritma 4: Aitlik Değerine Göre Sıralı Sezgisel Yaklaşım*

Bir elemanın kaliteye katkısı kümeye dâhil olduğunda veya olmadığında farklı hesaplanır. Bu iki değerin küçük olanına göre sıralama yapılır ve yerleştirilir.

$K = \left\lceil \frac{n}{m} \right\rceil$;
obje_listesi = Tüm objelerin kümesi;
**for** $j = 1$ $to$ $m$ **begin**
/* kutular için döngü */
Sıralı = obje_listesini $\min\{1 - R_3(i,j), R_4(i,j)\}$ değerine göre azalan sırala;
**foreach** i in Sıralı **begin**
**if** $(Kutu(j)$ $eleman$ $sayısı \geq K)$ **break**;
$i$ elemanı $Kutu(j)$ kutusuna yerleştir;
**endforeach**
**endfor**.

### *5.3.5 Algoritma 5: İki Aşamalı Rastgele Arama Algoritması*

Matrislerin detaylı doldurulmadığı veya çelişkilerin bulunduğu durumlarda rastgele yerleşimlerin kaliteleri hesaplanarak en iyi kalite değerini veren yerleşim sonuç olarak alınmaktadır. İterasyon sayısı, eleman sayısı ve kutu sayısına göre belirlenmelidir. Diğer algoritmalardan yavaş çalışsa da daha iyi değerler verebilmektedir.

$K = \left\lceil \frac{n}{m} \right\rceil$;
m = 1;
**for** i = 1 to n **begin**/* rastgele bir çözüm üret */
$Kutular(m)$ $içine$ $i.objeyi$ $yerleştir$;

**if** ( $Kutular(m)\ eleman\ sayısı \geq K$ ) $m = m + 1$;

**endfor**

// rand = 0 ile 1 arasında rastgele bir ondalık değer alır.

en_iyi_deger = 0;

**for** $m = 1\ to\ Iterasyon\_Sınırı$ **begin**

**if** ( $rand\ >\ 0.5$ ) **begin**

**for** k = 1 to kutu_sayisi **begin**

*Farklı iki kutunun sırasını değiştir;

**endfor**

**else**

**for** k = 1 to obje_sayısı **begin**

* Farklı iki kutudan seçilen 2 objenin yerini değiştir;

**endfor**

**endif**

*Derece = şu an ki çözümün değerini hesapla;

**if** ( $Derece \geq\ en\_iyi\_deger$) **begin**

en_iyi_deger = Derece;

Çözümü sakla;

**endif**

**endfor**.

Algoritmaların hepsi $A_1$ değeri hesaplanması düşünülerek hazırlanmıştır. Diğer durumların en zorlu hali $A_1$ değeri olduğu için algoritmalar diğer durumlar içinde kullanılabilir.

### *5.3.6 Karşılaştırmalar*

20 obje ve 4 kutu üzerinden oluşturulan matrisler ile testler yapılmıştır. TQ değeri 0.8 alınmıştır. $f_3\ ve\ f_4$ değerlerinin 0 alındığı durumda ortaya çıkan matrislere 'Düzenli Matris' denilmiştir. $f_3 = 0.6$ ve $f_4 = 0.6$ alındığında oluşan matrislere ise 'Dağınık Matris' adı ile bahsedilmiştir.

Düzenli Matris kullanılarak yapılan test sonuç karşılaştırmaları

Tablo 5.1'de verilmiştir. Verilerde adı geçen "Model" $R_3$ ve $R_4$ matrisleri oluşturulurken kullanılan model dağılımın kalite derecesidir. Temel, R3 sıralı, R4 sıralı ve üyelik sıralı algoritmalar ise bahsedilen Algoritma 1, 2, 3 ve 4'tür. Rastgele ise Algoritma 5'tir. Alt kısmında verilen sayılar ise hangi iterasyonda o kalite derecesini bulduğunu göstermektedir. Model satırı ile Temel sütunun kesiştiği yerdeki %45 ifadesi ise Model'in kalite değerinin %45 Temel algoritmasının kalite değeridir anlamına gelmektedir. Kolaylıkla hangi algoritma hangisinin ne kadarı kadar kalite elde edebilmiş bu tablo yardımı ile gözlenmektedir.

Tablo 5.1 20 Obje ve 4 Kutu üzerine yapılan karşılaştırma

| **Algoritma** | | Model | Temel | R3 Sıralı | R4 Sıralı | Üyelik Sıralı | Rastgele 4000 | Rastgele 40000 |
|---|---|---|---|---|---|---|---|---|
| **Derece** | | **0,731** | **0,328** | **0,731** | **0,258** | **0,195** | **0,380** | **0,602** |
| Model | **0,731** | 100% | 45% | 100% | 35% | 27% | 52% | 82% |
| Temel | **0,328** | 223% | 100% | 223% | 79% | 59% | 116% | 184% |
| R3 Sıralı | **0,731** | 100% | 45% | 100% | 35% | 27% | 52% | 82% |
| R4 Sıralı | **0,258** | 283% | 127% | 283% | 100% | 76% | 147% | 233% |
| Üyelik Sıralı | **0,195** | 375% | 168% | 375% | 132% | 100% | 195% | 309% |
| Rast. (4000) | **0,380** | 192% | 86% | 192% | 68% | 51% | 100% | 158% |
| Rast. (40000) | **0,602** | 121% | 54% | 121% | 43% | 32% | 63% | 100% |

40 obje ve 6 kutu üzerinden arka arkaya 20 defa yapılan testin ortalaması Tablo 5.2'de verilmiştir. Testte Algoritma 5 için 40000 iterasyon yapılmıştır. Testlerde bulunan minimum değer ile maksimum değer arasında sonuçlar normalize edildi ve ortalamaları tabloda sunuldu.

Tablo 5.2 40 obje ve 6 kutu ile detaylı dağılım (20 örnek ortalaması)

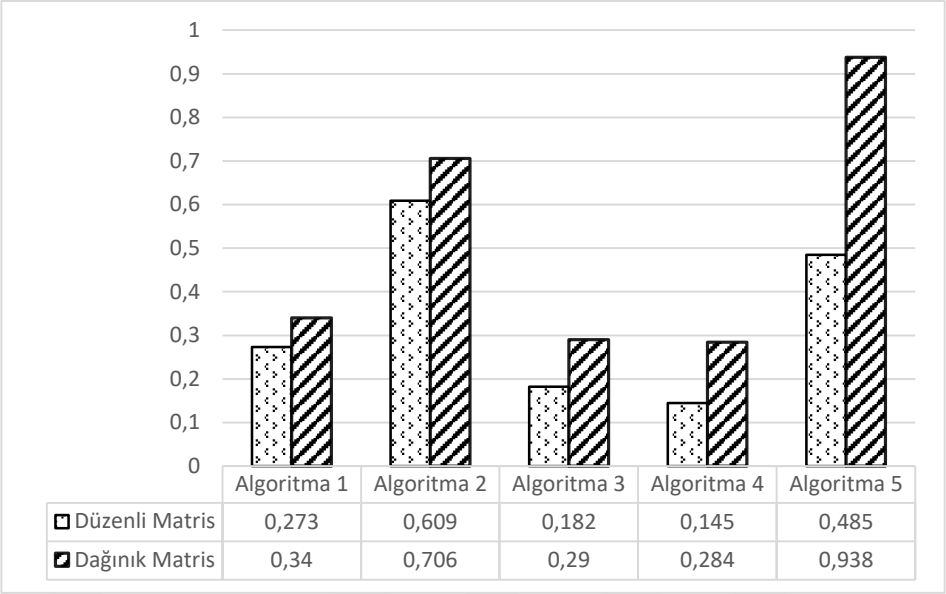
| **Algoritma** | | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|---|
| | **Derece** | **0,273** | **0,609** | **0,182** | **0,145** | **0,485** |
| 1 | **0,273** | 100% | 223% | 67% | 53% | 178% |
| 2 | **0,609** | 45% | 100% | 30% | 24% | 80% |
| 3 | **0,182** | 150% | 335% | 100% | 80% | 266% |
| 4 | **0,145** | 188% | 419% | 125% | 100% | 333% |
| 5 | **0,485** | 56% | 126% | 38% | 30% | 100% |

Dağınık matris kullanım sonuç karşılaştırması Tablo 5.3'te gösterilmektedir.

Tablo 5.3 40 obje ve 6 kutu ile gevşek dağılım (20 örnek ortalaması)

| **Algoritma** | | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|---|
| | **Derece** | **0,340** | **0,706** | **0,290** | **0,284** | **0,938** |
| 1 | **0,340** | 100% | 208% | 85% | 83% | 276% |
| 2 | **0,706** | 48% | 100% | 41% | 40% | 133% |
| 3 | **0,290** | 117% | 243% | 100% | 98% | 323% |
| 4 | **0,284** | 120% | 249% | 102% | 100% | 330% |
| 5 | **0,938** | 36% | 75% | 31% | 30% | 100% |

Bu iki dağılım halinde algoritmaların durumu Şekil 5.2'te grafik halinde gösterilmiştir. Rastgele arama algoritması olan algoritma 5, ortalama kalite değerlerini yüksek bulmuştur. $R_3$ değerlerine göre sıralama sonrası yerleştirme yapan algoritma 2 de yakın bir performans sergilemiştir. Burada algoritma 2'nin algoritma 5'ten çok daha hızlı çalıştığını belirtmek gerekir. Düzenli Matriste yüksek kalite değerine sahip grup sayısı çok çok daha az olduğu için algoritmalar dağınık matris örneklerinde elde ettikleri değerler kadar yüksek değerler elde edememeleri normaldir.




| | Algoritma 1 | Algoritma 2 | Algoritma 3 | Algoritma 4 | Algoritma 5 |
|---|---|---|---|---|---|
| Düzenli Matris | 0,273 | 0,609 | 0,182 | 0,145 | 0,485 |
| Dağınık Matris | 0,34 | 0,706 | 0,29 | 0,284 | 0,938 |

Şekil 5.2 Algoritmaların karşılaştırılması

Yapılan testler sonucu, 2. algoritma olarak önerilen $R_3$ matrisindeki verilere göre sıralama yapan algoritma, yüksek kalite değerleri bulmaktadır. Yerleştirme, her iki matris dağılımında da iyi sonuçlar verdiği gözlenmiştir. Algoritma 5, özellikle dağınık matris durumunda yüksek kalite değeri elde etmektedir ancak çalışma zamanı diğer algoritmalara göre oldukça yüksektir. Diğer algoritmalar ise benzer yakın değerler elde etmiş fakat kalite değeri olarak Algoritma 2 ve Algoritma 5'in gerisinde kalmıştır.

## 5.4 Kapasite Olarak Kullanılan Tüm Bulanık Sayılar İçin Çözüm

İlk yaklaşımların iki önemli dezavantajı vardı. Birincisi, takımların kapasitesinin homojen olması her zaman istenen durum olmayabilir. Örneğin, öğretmelerin, danışmanlık yapmak istediği öğrenci sayısı değişebilir. Kimi öğretmen 10, kimisi 2 öğrenci kabul edebilir. Özel durumu olan bir öğretmen danışmanlık yapmak istemeyebilir. Ancak ilk çözüm önerilerinde tercih edilen; takımların homojen dağılmasıydı. Bunun sağlanması için seçtiğimiz bulanık sayı $\overline{N} = [\,a, a, b, b]; a = K - 1, b = K; K = \left\lceil \frac{n}{m} \right\rceil$ şeklindeydi. Bu bulanık sayı az önce verilen 'danışman öğretmen' örneğini gerçekleştirmekten epey uzaktır. İkinci dezavantajı, problemlerin hacimleri ile ilgilidir ve bir sonraki başlıkta bahsedilecektir.

### *5.4.1 Maksimum N Değerini Bulmak İçin İterasyon Yöntemi*

Her dış bükey bulanık sayı ile çalışabilmesi için bir iterasyon içinde en yüksek N değeri elde edilmesi amaçlanmıştır. Algoritma, iki bölüme ayırılacaktır. İlk bölüm N değerini belli bir değerde tutacak ve ikinci bölüm bu N değeri altında belirlenen takım eleman sayıları sınırları altında en kaliteli dağılımı yapmaya çalışıp A değerini elde edecektir. Bir örnekle anlatmak için yamuksal bir bulanık sayı ele alalım.

$$\bar{L} = [10,20,30,40] \tag{5.11}$$

$\bar{L}$ bulanık sayısı en yüksek kalite değeri olan 1 değerini aldığı aralık [20,30] aralığıdır. Oluşacak takımlar minimum 20, maksimum 30 kişiden oluşması gerekir. İkinci kısıma bu aralık parametre olarak gönderilir ve ikinci kısım bu kısıtlar altında dağılımı gerçekleştir ve bu kısıtlar altında bulduğu A değerini döner. Bu değer 0.5 olsun. O halde ilk iterasyon sonucu N değeri 1 ve A değeri 0.5, toplam kalite ise 0.5 olarak hesaplanmıştır. Alt kalite olarak 0.5 atanır. Böylelikle A değeri için iterasyon 0.5 değerine kadar devam edecektir, daha küçük sayıları değerlendirmek gereksizdir. Bir sonraki iterasyonda N değeri bir miktar azaltılır. Örnekte bunu 0.9 olarak alalım. N = 0.9 için aralık [19,31] olur ve ikinci adıma bu parametreler ile geçilir. İkinci adım A değerini eğer 0.5'ten yüksek verirse alt kalite yeniden atanır, eğer değilse direk bir sonraki iterasyona geçilir. Bu döngü $N \leq alt\ kalite$ şartı sağlanana kadar devam eder. İterasyon bittiğinde son elde edilen alt kalite dağılımın genel kalitesi olacaktır. Algoritmayı şu şekilde ifade edebiliriz;

AltKalite = 0;
**for** $N = 1\ to\ AltKalite$ **step** $(-0.1)$ **begin**
    min = $\bar{N}_{left}(N)$;
    max = $\bar{N}_{right}(N)$;
    /*min-max değerleri bir önceki iterasyon ile aynı olabilir
        bir kontrol ile denenmiş aralıklar atlanmalıdır.*/
    $A_{tmp}$ = IkinciAsama(min, max);
    B = min$\{A_{tmp}, N\}$;

**if** ( $B > AltKalite$ ) **begin**

$AltKalite = B$;

*dağılımı ve kaliteyi sakla.

**endif**

**endfor**.

Verilen algoritma problemin çözümünde temel çalışacak döngüdür. Önceki bölümdeki algoritmaların, bu iterasyon yöntemi ile çalışabilmesi için bazı güncellemelerin yapılması gerekmiştir.

### *5.4.2 Temel Algoritmanın (Algoritma 1) Güncellenmesi*

Daha önceden sunulan; Algoritma 1, Nasibov'un sunduğu algoritmanın bu probleme uyarlanması ile oluşturuldu. Bu algoritma, yüksek kalite değerinden başlayıp kutulara o kalite değerinde kalmak şartı ile bir yerleşim bulmaya çalışmaktadır. Eğer bulamazsa, kalite değeri bir miktar azaltılır ve yeni değer ile tekrar denenir, bir yerleşim bulduğunda ise algoritma sonlanmaktadır. Bu algoritmanın baz algoritma olması sebebi ile üzerinde bu fikri bozacak bir değişikliğe gidilmemiştir. Eğer sağlanan dağılımda gruplar dağılım sonunda istenen maks-min aralığında değilse kalite değeri düşürülüp tekrar denenmesi sağlanmıştır.

### *5.4.3 Algoritma 2 ve 3'ün Güncellenmesi*

Algoritma 2, küçük hacimli ve yakın eleman sayılarına sahip takım dağılımları için kabul edilebilir sonuçlar vermişti. İterasyon algoritmasının, parametre olarak gönderdiği min-max değerleri dikkate alınacak şekilde güncellenmiştir. Algoritma şu şekildedir;

R = Algoritma 2 ise $R_3$, değilse $R_4$

ogr = tüm öğrenciler listesi;

**for** $j = 1\ to\ m$ **begin**

sirali = ogr listesini $R$ azalan/artan sırala;

**for** $i = 1\ to\ sayi(\ sirali\ )$ **begin**

**if** ( $sayi\ (\ takim_j\ ) > min$ ) **break**;

*$sirali_i$ 'yi $takim_j$ 'ye yerleştir.

*ogr listesinden $sirali_i$ 'yi çıkar.

**endfor**

**endfor**

**if** ( $sayi\ (\ ogr\ ) == 0$ ) **return**;

**for** $i = 1\ to\ sayi(\ ogr\ )$ **begin**

**for** $j = 1\ to\ m$ **begin**

**if** ( $sayi\ (\ takim_j\ ) > max$ ) **continue**;

***$ogr_i$ 'yi $takim_j$ 'ye yerleştir.

**break**;

**endfor**

**endfor**.

## 5.5 Büyük Hacimli Problemler İçin Çözüm Önerisi

Önceki bölümde bahsedilen dezavantajdan ikincisi ise verilen sezgisel algoritmaların çok kişi ve takım içeren problem kümeleri üzerinde verimsiz çalışmasıdır. Verimli çalışan bir algoritma için problemi daha derin incelemek gerekmektedir. Yapılan atamada A kalitesini belirleyen $R_3$ ve $R_4$ matrislerindeki verilerdir.

$$R_K = \begin{bmatrix} r_{11} & \cdots & r_{1\mathrm{m}} \\ \vdots & \ddots & \vdots \\ r_{n1} & \cdots & r_{nm} \end{bmatrix}, \tag{5.12}$$

$$r_{ij} = \min\{R_4(i,j), \min_{\substack{m=1 \\ m \neq j}}(1 - R_3(i,m))\}, i = 1 \dots n\ , j = 1 \dots m \tag{5.13}$$

$R_K$ matrisi, $R_3$ ve $R_4$ matrislerinin aynı anda yorumlanmasını sağlayan bir matristir. Matrisin satırları takımları, sütunları ise kişileri temsil eder. Kesiştikleri yerdeki değer ise kişinin o takıma girmesi halinde A kalitesine katkısını göstermektedir. Tablo

5.4'de verilen R matrisleri üzerinden önce $R_K$ matrisi oluşturulacak sonra elenen kalite değerlerinin nasıl elendiği gösterilecektir.

Tablo 5.4 Örnek çözümü için kullanılacak R matrisleri

| | $R_3$ | | | | $R_4$ | | | |
|---|---|---|---|---|---|---|---|---|
| | T1 | T2 | T3 | T4 | T1 | T2 | T3 | T4 |
| 1 | 0,04 | 0,01 | 0,09 | 0,70 | 0,23 | 0,36 | 0,25 | 0,85 |
| 2 | 0,70 | 0,20 | 0,02 | 0,10 | 0,96 | 0,14 | 0,16 | 0,29 |
| 3 | 0,15 | 0,18 | 0,73 | 0,14 | 0,40 | 0,23 | 0,87 | 0,37 |
| 4 | 0,17 | 0,14 | 0,65 | 0,13 | 0,31 | 0,26 | 0,94 | 0,29 |
| 5 | 0,13 | 0,77 | 0,15 | 0,10 | 0,20 | 0,86 | 0,41 | 0,13 |
| 6 | 0,10 | 0,78 | 0,11 | 0,14 | 0,13 | 0,81 | 0,25 | 0,33 |
| 7 | 0,16 | 0,75 | 0,06 | 0,06 | 0,35 | 0,95 | 0,21 | 0,16 |
| 8 | 0,66 | 0,10 | 0,12 | 0,17 | 0,96 | 0,26 | 0,28 | 0,31 |
| 9 | 0,61 | 0,13 | 0,18 | 0,02 | 0,95 | 0,19 | 0,33 | 0,18 |
| 10 | 0,20 | 0,17 | 0,13 | 0,74 | 0,11 | 0,19 | 0,42 | 0,89 |
| 11 | 0,12 | 0,10 | 0,11 | 0,15 | 0,36 | 0,32 | 0,76 | 0,81 |

Verilen matrislerde ki değerlerden $R_K$ matrisini oluşturmak için $R_4$ ve $1 - R_3$ matrisi gerek duyulur. (5.3)'e göre bir kişi takıma dahil ise $R_4$ , değilse $1 - R_3$ değeri kullanılacaktır. Tablo 5.5'te görüldüğü gibi $R_K$ matrisinde 1. kişi 4. takıma dahil olursa kaliteye katkısı 0,85 olarak hesaplanmıştır. Koyu karakterle de belirtildiği gibi 0,85 değeri $R_4$ matrisinden alınan 0.85 değeri ve $1 - R_3$ matrisinden gelen 0.96, 0.99, 0.91 değerlerinin en küçüğüdür. 2. kişi içinde yine bakılırsa $R_K$ matrisinde 2. takımda olduğu durumda gelen kalite değeri, $R_4$ matrisinde 0.14 ve $1 - R_3$ matrisinden 0.30, 0.99, 0.90 değerlerinin en küçüğü 0.14'tür.

Tablo 5.5 $R_K$ matrisinin hesaplanması

| | $R_4$ | | | | $1 - R_3$ | | | | $R_K$ | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | T1 | T2 | T3 | T4 | T1 | T2 | T3 | T4 | T1 | T2 | T3 | T4 |
| 1 | 0,23 | 0,36 | 0,25 | **0,85** | **0,96** | **0,99** | **0,91** | 0,30 | 0,23 | 0,30 | 0,25 | **0,85** |
| 2 | 0,96 | **0,14** | 0,16 | 0,29 | **0,30** | 0,80 | **0,99** | **0,90** | 0,80 | **0,14** | 0,16 | 0,29 |
| 3 | 0,40 | 0,23 | 0,87 | 0,37 | 0,85 | 0,82 | 0,27 | 0,86 | 0,27 | 0,23 | 0,82 | 0,27 |
| 4 | 0,31 | 0,26 | 0,94 | 0,29 | 0,83 | 0,86 | 0,36 | 0,88 | 0,31 | 0,26 | 0,83 | 0,29 |
| 5 | 0,20 | 0,86 | 0,41 | 0,13 | 0,87 | 0,23 | 0,85 | 0,90 | 0,20 | 0,85 | 0,23 | 0,13 |
| 6 | 0,13 | 0,81 | 0,25 | 0,33 | 0,90 | 0,22 | 0,90 | 0,86 | 0,13 | 0,81 | 0,22 | 0,22 |
| 7 | 0,35 | 0,95 | 0,21 | 0,16 | 0,84 | 0,25 | 0,94 | 0,94 | 0,25 | 0,84 | 0,21 | 0,16 |
| 8 | 0,96 | 0,26 | 0,28 | 0,31 | 0,34 | 0,90 | 0,88 | 0,83 | 0,83 | 0,26 | 0,28 | 0,31 |
| 9 | 0,95 | 0,19 | 0,33 | 0,18 | 0,39 | 0,87 | 0,82 | 0,98 | 0,82 | 0,19 | 0,33 | 0,18 |
| 10 | 0,11 | 0,19 | 0,42 | 0,89 | 0,81 | 0,83 | 0,87 | 0,26 | 0,11 | 0,19 | 0,26 | 0,81 |
| 11 | 0,36 | 0,32 | 0,76 | 0,81 | 0,88 | 0,90 | 0,89 | 0,85 | 0,36 | 0,32 | 0,76 | 0,81 |

$R_K$ üzerinde birinci kişi için elde edilen değerlere daha yakından bakıldığında, bu kişinin *A* değerine katkısı, birinci, ikinci ve üçüncü takıma girdiğinde 0.30 derecenin üzerine çıkamamaktadır. Dördüncü takımda ise katkısı 0.85 'tir. Bu kişinin, kaliteye katkısı düşük takımlara girmesi istenmeyen bir durumdur. Bir sonraki adım olarak bu düşük kaliteler $R_K$ matrisinden çıkarılacaktır. Bunun için tüm sütunları geçici olarak büyükten küçüğe sıralanır. Ardışık kaliteler arasındaki farkın en büyük olduğu yer belirlenir. Bu fark belirlenirken bir 'kesme eşiği' kullanılacaktır. Fark, eşiğin altında kalıyorsa dikkate alınmayacaktır. Bunun nedeni örnekle daha iyi anlaşılır. Bir kişinin takımlara dağıldığında katkısı 0.99 – 0.98 – 0.97 – 0.96 şeklinde olduğunda eğer bir eşik belirlenmezse 0.99 kalite değerinden sonraki değerler çıkarılır. Oysaki bu değerler yüksek değerlerdir ve o öğrencinin serbest bir şekilde istenilen takıma yerleştirilebilmesine olanak tanır. Başka öğrencilerden mecburi bir takıma konma zorunluluğu çıktığında bu serbest öğrenciler *N* kalite değerini yükseltmek adına ihtiyaç olan herhangi bir takıma yerleştirilebilir. Yazının ilerleyen kısımlarında, testlerde bu 'kesme eşiği' değeri 0.3 olarak alınacaktır. Tablo 5.6'da $R_K$ matrisinin sıralanmış ve sonra kötü değerlerin çıkarılmış hali görülmektedir. Bu hali daha sonra $R_K^*$ şeklinde isimlendirilir.

Tablo 5.6 $R_K$ matrisinin son halinin elde edilmesi

| | $R_K$ | | | | Satır sıralı $R_K$ | | | | $R_K$ son hali = $R_K^*$ | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | T1 | T2 | T3 | T4 | T1 | T2 | T3 | T4 | T1 | T2 | T3 | T4 |
| 1 | 0,23 | 0,30 | 0,25 | 0,85 | 0,85 | ~~0,30~~ | ~~0,25~~ | ~~0,23~~ | 0,00 | 0,00 | 0,00 | **0,85** |
| 2 | 0,80 | 0,14 | 0,16 | 0,29 | 0,80 | ~~0,29~~ | ~~0,16~~ | ~~0,14~~ | **0,80** | 0,00 | 0,00 | 0,00 |
| 3 | 0,27 | 0,23 | 0,82 | 0,27 | 0,82 | ~~0,27~~ | ~~0,27~~ | ~~0,23~~ | 0,00 | 0,00 | **0,82** | 0,00 |
| 4 | 0,31 | 0,26 | 0,83 | 0,29 | 0,83 | ~~0,31~~ | ~~0,29~~ | ~~0,26~~ | 0,00 | 0,00 | **0,83** | 0,00 |
| 5 | 0,20 | 0,85 | 0,23 | 0,13 | 0,85 | ~~0,23~~ | ~~0,20~~ | ~~0,13~~ | 0,00 | **0,85** | 0,00 | 0,00 |
| 6 | 0,13 | 0,81 | 0,22 | 0,22 | 0,81 | ~~0,22~~ | ~~0,22~~ | ~~0,13~~ | 0,00 | **0,81** | 0,00 | 0,00 |
| 7 | 0,25 | 0,84 | 0,21 | 0,16 | 0,84 | ~~0,25~~ | ~~0,21~~ | ~~0,16~~ | 0,00 | **0,84** | 0,00 | 0,00 |
| 8 | 0,83 | 0,26 | 0,28 | 0,31 | 0,83 | ~~0,31~~ | ~~0,28~~ | ~~0,26~~ | **0,83** | 0,00 | 0,00 | 0,00 |
| 9 | 0,82 | 0,19 | 0,33 | 0,18 | 0,82 | ~~0,33~~ | ~~0,19~~ | ~~0,18~~ | **0,82** | 0,00 | 0,00 | 0,00 |
| 10 | 0,11 | 0,19 | 0,26 | 0,81 | 0,81 | ~~0,26~~ | ~~0,19~~ | ~~0,11~~ | 0,00 | 0,00 | 0,00 | **0,81** |
| 11 | 0,36 | 0,32 | 0,76 | 0,81 | 0,81 | 0,76 | ~~0,36~~ | ~~0,32~~ | 0,00 | 0,00 | 0,76 | **0,81** |

Tablo 5.6'daki $R_K^*$ matrisi elde edildiğinde 0'dan farklı elde edilen en küçük değer her zaman A kalite değerine eşit ya da yüksek olacaktır. Kişilerin satırlarında bulunan koyu işaretli rakamlar ait oldukları kolondaki takıma atanırsa da $A = 0.8$ olarak bulunur.

### *5.5.1 Algoritma 6: Macar Algoritması İle Çözüm*

$R_K^*$ matrisi elde edildikten sonra bu matris üzerindeki değerlere göre öğrencilerin takımlara atanması gerekmektedir. Burada bu dağılımı sağlamak için ilk macar algoritması kullanılmıştır. Macar algoritmasını (Kuhn, 2009) adım algoritması olarak şu şekildedir;

m: satır sayısı, girdi;

n: sütun sayısı, girdi;

$C_{ij}$: atama yapılacak verilerin matris formu, girdi;

**Adım 1**: m != n ise yapay satır ve/veya sütun ekleyerek m = n yap.

**Adım 2**: Problem en büyükleme ise atama tablosundaki en büyük $C_{ij}$'den, tüm $C_{ij}$ 'leri çıkar.

**Adım 3**: Satır indirgeme. Her satırdaki en küçük $C_{ij}$'yi o satırın tümünden çıkar.

**Adım 4**: Sütun indirgeme. Her sütundaki en küçük $C_{ij}$'yi o sütunun tümünden çıkar.

**Adım 5**: Atama tablosundaki tüm sıfırları örtecek şekilde en az sayıda yatay veya dikey çizgi çiz.

**Adım 6**: Çizgi sayısı < m = n ise **Adım 7**'ye git. Çizgi sayısı = m = n ise en iyi çözüm bu tabloda mevcuttur.

**Adım 7:** Çizgilerle örtülmemiş elemanların en küçüğünü bul, bu değeri örtülmemiş tüm değerlerden çıkar. Çizgilerin kesiştiği değerlere ekle ve **Adım 5**'e git.

Tablo 5.7'de görüldüğü gibi matrisin önce kare matrise çevrilmesi gerekmektedir. Bunun için ilk adımda takım sayılarını ifade eden sütun sayıları katları olacak şekilde arttırılır. 11 öğrenciyi geçmesi için 4 takım, 3 katı olan 12 sütun ile ifade edilmiştir. Sütunlar arttırıldığında değerler aynen kopyalanıp diğer sütunlara takımlara dikkat edilerek yazılır. Daha sonra ki adımda kare olması için öğrenci sayısına tam kare olmasına yetecek kadar satır eklenir ki bu örnekte bu 1 satırdır. Bu satır 12. satır olarak eklenir bu satıra tüm değerler 0 olarak yazılır. Bu bölüm algoritmanın 1. adımıdır.

Tablo 5.7 Macar algoritması için $R_K^*$ matrisi kare matrise çevirme

| | T1 | T2 | T3 | T4 | T1 | T2 | T3 | T4 | T1 | T2 | T3 | T4 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 0,00 | 0,00 | 0,00 | 0,85 | 0,00 | 0,00 | 0,00 | 0,85 | 0,00 | 0,00 | 0,00 | 0,85 |
| 2 | 0,80 | 0,00 | 0,00 | 0,00 | 0,80 | 0,00 | 0,00 | 0,00 | 0,80 | 0,00 | 0,00 | 0,00 |
| 3 | 0,00 | 0,00 | 0,82 | 0,00 | 0,00 | 0,00 | 0,82 | 0,00 | 0,00 | 0,00 | 0,82 | 0,00 |
| 4 | 0,00 | 0,00 | 0,83 | 0,00 | 0,00 | 0,00 | 0,83 | 0,00 | 0,00 | 0,00 | 0,83 | 0,00 |
| 5 | 0,00 | 0,85 | 0,00 | 0,00 | 0,00 | 0,85 | 0,00 | 0,00 | 0,00 | 0,85 | 0,00 | 0,00 |
| 6 | 0,00 | 0,81 | 0,00 | 0,00 | 0,00 | 0,81 | 0,00 | 0,00 | 0,00 | 0,81 | 0,00 | 0,00 |
| 7 | 0,00 | 0,84 | 0,00 | 0,00 | 0,00 | 0,84 | 0,00 | 0,00 | 0,00 | 0,84 | 0,00 | 0,00 |
| 8 | 0,83 | 0,00 | 0,00 | 0,00 | 0,83 | 0,00 | 0,00 | 0,00 | 0,83 | 0,00 | 0,00 | 0,00 |
| 9 | 0,82 | 0,00 | 0,00 | 0,00 | 0,82 | 0,00 | 0,00 | 0,00 | 0,82 | 0,00 | 0,00 | 0,00 |
| 10 | 0,00 | 0,00 | 0,00 | 0,81 | 0,00 | 0,00 | 0,00 | 0,81 | 0,00 | 0,00 | 0,00 | 0,81 |
| 11 | 0,00 | 0,00 | 0,76 | 0,81 | 0,00 | 0,00 | 0,76 | 0,81 | 0,00 | 0,00 | 0,76 | 0,81 |
| 12 | 0,00 | 0,00 | 0,00 | 0,00 | 0,00 | 0,00 | 0,00 | 0,00 | 0,00 | 0,00 | 0,00 | 0,00 |

Kare matrise çevrilen matris henüz Macar algoritması için hazır değildir. Hazır olması için 2 düzeltme daha yapılacaktır. İlki pozitif olan değerlerin hepsi negatife çevrilir. Amacı minimize etmek olan Macar algoritması için bu gereklidir. İkinci ise 0

değerleri mümkün olan büyük bir pozitif sayı yapmaktır. Küme büyüdükçe bu sayıda büyük olmalı, örneğin $10^6$ gibi bir değer alınabilir. Bunun nedeni ise çok büyük kümelerde küsüratlar birleşik negatif bir sayı yerine 0 değerini seçebiliyor olmasıdır. Bu da algoritmanın 2. adımı yerine kullanılmıştır. Tablo 5.8 'de Macar algoritması için hazır matris verilmiştir.

Tablo 5.8 Macar algoritmasına hazır $R_K^*$ kare matrisi

| | T1 | T2 | T3 | T4 | T1 | T2 | T3 | T4 | T1 | T2 | T3 | T4 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | $10^6$ | $10^6$ | $10^6$ | -0,85 | $10^6$ | $10^6$ | $10^6$ | -0,85 | $10^6$ | $10^6$ | $10^6$ | -0,85 |
| 2 | -0,80 | $10^6$ | $10^6$ | $10^6$ | -0,80 | $10^6$ | $10^6$ | $10^6$ | -0,80 | $10^6$ | $10^6$ | $10^6$ |
| 3 | $10^6$ | $10^6$ | -0,82 | $10^6$ | $10^6$ | $10^6$ | -0,82 | $10^6$ | $10^6$ | $10^6$ | -0,82 | $10^6$ |
| 4 | $10^6$ | $10^6$ | -0,83 | $10^6$ | $10^6$ | $10^6$ | -0,83 | $10^6$ | $10^6$ | $10^6$ | -0,83 | $10^6$ |
| 5 | $10^6$ | -0,85 | $10^6$ | $10^6$ | $10^6$ | -0,85 | $10^6$ | $10^6$ | $10^6$ | -0,85 | $10^6$ | $10^6$ |
| 6 | $10^6$ | -0,81 | $10^6$ | $10^6$ | $10^6$ | -0,81 | $10^6$ | $10^6$ | $10^6$ | -0,81 | $10^6$ | $10^6$ |
| 7 | $10^6$ | -0,84 | $10^6$ | $10^6$ | $10^6$ | -0,84 | $10^6$ | $10^6$ | $10^6$ | -0,84 | $10^6$ | $10^6$ |
| 8 | -0,83 | $10^6$ | $10^6$ | $10^6$ | -0,83 | $10^6$ | $10^6$ | $10^6$ | -0,83 | $10^6$ | $10^6$ | $10^6$ |
| 9 | -0,82 | $10^6$ | $10^6$ | $10^6$ | -0,82 | $10^6$ | $10^6$ | $10^6$ | -0,82 | $10^6$ | $10^6$ | $10^6$ |
| 10 | $10^6$ | $10^6$ | $10^6$ | -0,81 | $10^6$ | $10^6$ | $10^6$ | -0,81 | $10^6$ | $10^6$ | $10^6$ | -0,81 |
| 11 | $10^6$ | $10^6$ | -0,76 | -0,81 | $10^6$ | $10^6$ | -0,76 | -0,81 | $10^6$ | $10^6$ | -0,76 | -0,81 |
| 12 | $10^6$ | $10^6$ | $10^6$ | $10^6$ | $10^6$ | $10^6$ | $10^6$ | $10^6$ | $10^6$ | $10^6$ | $10^6$ | $10^6$ |

Algoritma şu şekildedir;

**Adım 1**. verilen $R_3$ ve $R_4$ matrisinden $R_K^*$ matrisini oluştur.

**Adım 2**. $n$: kişi sayısı, $m$: takım sayısı

**Adım 3**. min, max: algoritmanın 1 aşamasından gönderilecek parametre

**/*** macar algoritması 1'e 1 atama yaptığı için
bir takımda olabilecek maksimum eleman sayısı kadar
büyük bir kare matris olmalıdır, d bu matrisin boyutudur. ***/**

**Adım 4**. d = m * max;

**Adım 5**. $R_K^*$ matrisini $d \times d$ boyutlarına verilen kurallar ile genişlet.

**Adım 6**. $R_K^*$ matrisine macar algoritmasını uygula (Stern, 2014). (Tablo 4-6,4-8)

**Adım 7**. Çıkan sonuca göre dağılımı gerçekleştir.

### *5.5.2 Algoritma 7: Bv3 Algoritması*

Bv3 algoritması olarak adlandırılan algoritmanın çalışması için $R_K^*$ matrisine ihtiyaç duyar. Algoritma açgözlü bir yaklaşım izler. Kısaca algoritma bir döngü içinde en az takıma girebilen kişiyi, içine girdiğinde en yüksek kalite değerini verecek takıma yerleştirmesi şeklinde devam eder. Sonra takımları teker teker ele alır. Ele aldığı takımın eleman sayısı maksimum eleman sayısından fazlaysa takımdaki elemanlar derecesi azalacak şekilde sıralanır ve sırayla elemanlar yerleştiğinde en yüksek dereceyi alacağı sonraki takıma atanır. Bu atama esnasında atanacak takımın maksimum eleman sayısını aşmama şartına bakılır. Daha sonra minimum eleman sayısı altında kalan takımlara bakılır. Bu takımlara girebilecek öğrenciler derecelerine göre sıralanır, eğer o öğrenci bu takıma atıldığında önceki takım minimum öğrenci sayısı altında kalmıyorsa bu takıma atanır. Bu işlemler sonucunda tüm takımlar minimum ve maksimum öğrenci sayısı arasında kalır. Bu tezde kullanılan tüm algoritmalarda C# 6.0 kullanıldı, özellikle bu algoritmada C#'ın LinQ uzantıları, anlatılan tüm şartları sağlayan sorguları yapmayı kolaylaştırmıştır.

Algoritma şu şekildedir;

Verilen $R_3$ ve $R_4$ matrisinden $R_K^*$ matrisini oluştur.
/* kişilerin kaç takıma girebildiğini ifade eder */
$\gamma$ = $R_K^*$ matrisine göre satırlarında kaç adet 0'dan büyük değer olduğunu tutar.
$\delta$= $R_K^*$ matrisini ifade eder.
**while** ( $true$ ) **begin**
    s = $\gamma$ listesindeki en az değere sahip kişiyi listeden al;
    liste = $\delta_s$ listesinde 0'dan büyük derecelerin azalan sıralaması
    **for** $i = 1$ $to$ $sayi$ ( $liste$ ) **begin**
        takım = $liste_i$; //sıralamada i. takım
        **if** ($sayi$ ( $takım$ ) > maximum ) **continue**;
        *s'yi takıma yerleştir;
        **break**;
    **endfor**

**if** ( $s\ bir\ takıma\ yerleşmediyse$ ) **begin**

//eğer liste boşsa bir çözüm yoktur.

//daha farklı bir kesme eşiği ile $R_K^*$ matrisi oluşturulmalıdır.

*****liste'deki ilk takıma yerleştir.

**endif**

**endwhile**

liste_takım = tüm takımlar içinde maximum değerden yüksek takımlar;

**for** $i = 1\ to\ sayı\ (liste\_takım)$ **begin**

takım = $liste\_takım_i$;

ogr_liste = takım içindeki kişileri dereceleri azalan şekilde sırala;

**for** $j = 1\ to\ sayı(ogr\_liste)$ **begin**

ogr = $ogr\_liste_j$;

/* en uygun: derecesi en yüksek, eleman sayısı < maksimum */

yeni_takım = ogr'nin yerleşebileceği en uygun takım;

ogr'yi yeni_takıma gönder.

**if** ( $sayı\ (takım) < maksimum$) **break**;

**endfor**

**endfor**

liste_takım = tüm takımlar içinde minimum değerin altında elemana sahip olanlar;

**for** $i = 1\ to\ sayı\ (liste\_takım)$ **begin**

takım = $liste\_takım_i$;

/* takıma uygun: derecesi yüksek,

çıktığı takımın eleman sayısı > minimum */

ogr_liste = tüm öğrenciler içinde takıma uygun olanlar;

ogr_liste içinden takımın eleman sayısını minimum üstüne

çıkaracak kadar elemanı takıma gönder;

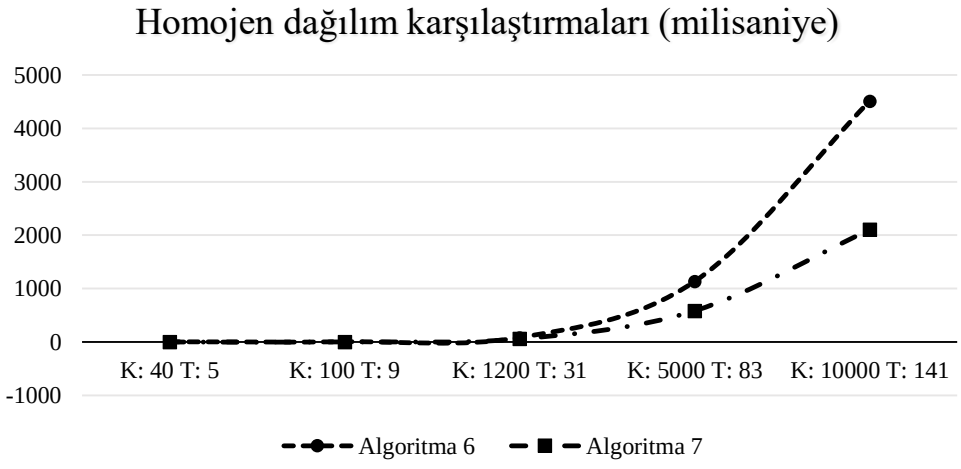
**endfor**.

### *5.5.3 Karşılaştırmalar*

Farklı eleman ve grup sayılarına ait yapılan test sonuçlarına göre algoritma 6 ve algoritma 7 oldukça istikrarlı çalışmaktadır. Testlerde kullanılan problem kümeleri

önceki bölümde verilen algoritma ile üretilmiştir. Testler; C# 6.0 ile ara yüzü olmadan konsol ortamında yazılmış bir program ile yapılmıştır. Program Visual Studio 2015 Community IDE 'si ile geliştirilmiştir. Testler; i7 4712MQ CPU ve 8 GB RAM ile Windows 10 Home işletim sistemi üzerinde derlenmiş ve çalıştırılmıştır. Program 64bit mimarisi için derlenmiştir. 32bit mimaride yüksek hacimli problemler 2GB RAM kısıtı nedeniyle çalışmayabilmektedir.

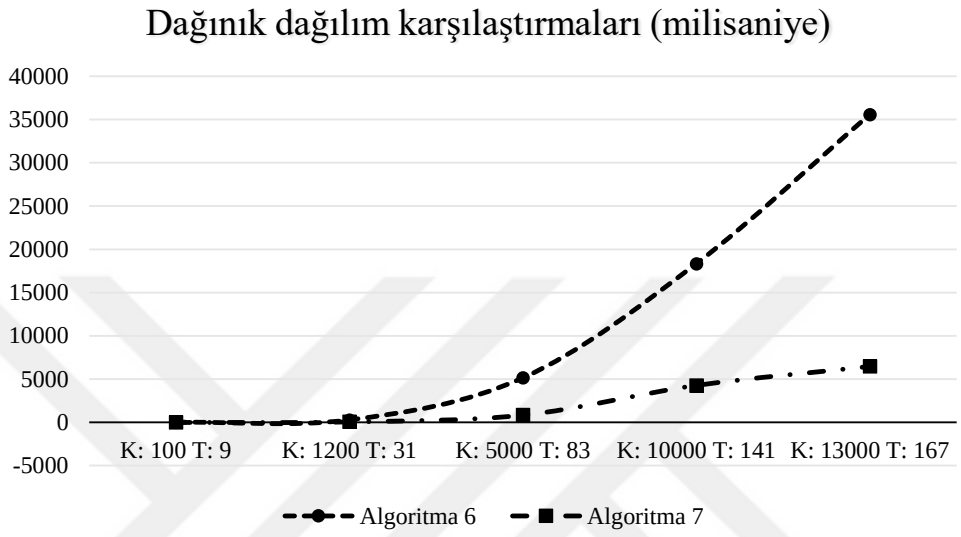
Yapılan testlerde üretilen problem kümelerin hacimleri, matrislerin esneklikleri ne kadar değişsede Algoritma 6 ve 7 başarılı sonuçlar vermiştir. Dolayısı ile bu algoritmaları karşılaştırırken, başarılarından ziyade süre performansları karşılaştırılacaktır. Şekil 5.3'te 40 kişi 5 takım, 100 kişi 9 takım, 1200 kişi 31 takım, 5000 kişi 83 takım, 10000 kişi ve 141 takım üzerinden yapılan testlerin her birinde algoritmaların ikiside *MD* kalite değerini yakalamaktadırlar. Şekilde algoritmaların milisaniye cinsinden cevabı verme süreleri gösterilmiştir. 10000 kişi ve 141 takıma sahip büyük ölçekli problem kümesinde Algoritma 6 problemi 4511ms'de Algoritma 7 ise 2103ms'de çözmüştür. 100 kişiye kadar olan testlerde her iki algoritmada 1ms'nin altında cevap vermektedir.




Şekil 5.3 Takımların homojen dağılımı durumunda hız karşılaştırmaları

Şekil 5.4'de algoritmaların dağınık oluşturulmuş R matrisleri üzerine çözüm performansları gösterilmiştir. Burada her iki algoritmada en iyi çözümü süre sonunda bulmaktadır. Bu test uygulanırken algoritmalar 13000+ elemana sahip problem

kümelerine zorlandı, ancak Algoritma 6'nın hafıza sınırına ulaşması sonucu çoğunlukla bir çözüm üretemedi. Son test olan 13000 kişi ve 167 takım barındıran problemi çözüm esnasında görev yöneticisinden takip edilen programın RAM kullanımı Algoritma 6 için 6 GB seviyesini gördü, Algoritma 7 ise 200 MB seviyesini aşmadı. Her iki algoritmada da veriler double veri tipinde saklanmaktadır.




Şekil 5.4 Takımların dağınık dağılımda hız karşılaştırmaları

# BÖLÜM ALTI
# SONUÇLAR

Bu tez çalışmasında, bulanık mantık kullanılarak en iyi proje takımlarının oluşturulması problemi ve çözüm algoritmaları ele alındı. Çalışma başlarında bazı basit sezgisel yaklaşımlar kullanıldı. Bu yaklaşımların bazıları, küçük problem kümelerinde kabul edilebilir sonuçlar vermiştir. Ancak daha büyük problem kümelerinde başarı sağlamak için farklı çözüm yolları aranmıştır. Algoritma 6 ve Algoritma 7 bu problemleri başarı ile çözmüştür. Çalışma sonunda gerçekleşenler şu şekildedir;

a- Problem modeli üzerine:

- Nasibov'un çalışmasında kullanılan matrislerden sadece $R_3$ ve $R_4$ matrisleri kullanıldı ve $S_{m+1}$ olarak tanımlanan kutu tamamen kaldırıldı. Böylelikle tüm kişiler, geçerli takımlara yerleşmesi sağlandı.
- Temel çalışmada kullanılan amaç fonksiyonu (min-min, $A_1$) yanına 3 tane daha amaç fonksiyonu ( min-max, max-min, max-max, $A_2$, $A_3$, $A_4$ ) tanımlandı ve gerçek hayattaki kullanım alanları açıklandı.
- Temel çalışmadan farklı olarak takımların eleman sayılarının da problem kalitesine katkısı önemlidir, takımların eleman dağılımı bir bulanık sayı ile ifade edildi ve dağılımın kalitesinin belirlenmesinde kullanıldı.

b- Problemin çözümü üzerine:

- Problemin çözümü için ilk aşamada temel algoritma bu problemi çözebilmesi için modifiye edildi. 3 tane bazı R matrislerindeki değerlere göre sıralama ve ilk uygun yere yerleştirme adımlarını uygulayan algoritma geliştirildi. Ayrıca, iterasyon temelli rastgele arama algoritması geliştirildi.
- Bu 5 algoritmanın eksikliklerini gidermek adına Macar algoritması ve açgözlü algoritma temelli yaklaşımlar üretildi. Sonuç itibari ile bu iki algoritmada son derece kaliteli dağılımlar elde edebildi.

c- Tezin çıktıları:

- Algoritma 1 ve 2'yi içeren bir çalışma, bildiri olarak '*5th International Conference on Advanced Technology & Sciences (ICAT'17)*' konferansa gönderilmiş ve sözlü sunum için kabul edilmiştir.
- Macar algoritmasını içeren Algoritma 6 üzerine bir makale dergiye gönderilmek üzere hazırlanmıştır.

**KAYNAKLAR**


Baets, B. ve Meyer, H. (2003). On the existence and construction of T-transitive cloures. *Information Sciences, 152*, 167-179.

Bansal, N., Lodi, A. ve Sviridenko, M. (2005). A tale of two dimensional bin packing. *46th Annual IEEE Symposium on Foundations of Computer Science (FOCS'05).* Pittsburgh: IEEE.

Chazelle, B. (1983). The bottomn-left bin-packing heuristic: an efficient implementation. *IEEE Transactions on Computers*, 697-707.

Garey, M. ve Jonhson, D. (1981). Approximation algorithms for bin packing problems: a survey. *Analysis and design of algorithms in combinatorial optimization* (s. 147-172). içinde Springer Vienna.

György, A., Lugosi, G. ve Ottucsak, G. (2010). On-line squential bin packing. *Journal of Machine Learning Research*, 89-109.

Jang, J.-S., Sun, C.-T. ve Mizutani, E. (1997). *Neuro-fuzzy and soft computing.* USA: Pearson.

Jylänki, J. (2010). *A thousand ways to pack the bin - a practical approach to two-dimensional rectangle bin packing. 3 Şubat 2017, http://clb.demon.fi/files/RectangleBinPack.pdf*

Kandel, A. ve Yelowitz, L. (1974). Fuzzy chains. *IEEE Transaction and. Systems, Man and Cybernetics, SMC-4*, 472-475.

Kim, J.-K., Kwang, H. ve Yoo, S. (2001). Fuzzy bin packing problem. *Fuzzy Sets and Systems 120*, 429-434.

Korf, R. (2002). A new algorithm for optimal bin packing. *In Proceedings of 18th AAAI*, 731–736.

Korf, R. E. (2003). An improved algorithm for optimal bin packing. *In Proceedings of the 18th International Joint Conference on Artificial Intelligence (IJCAI'03).* (s. 1252-1258). San Francisco: Morgan Kaufmann Publishers Inc.

Korte, B. ve Vygen, J. (2006). Bin-packing. B. Korte ve J. Vygen içinde, *Combinatorial optimization: theory and algorithms* (s. 426–441). Bonn: Springer.

Kuhn, H. (2009). The hungarian method for the assignment problem. H. Kuhn içinde, *50 Years of Integer Programming 1958-2008* (s. 29-47). Berlin: Springer Berlin Heidelberg.

Lodi, A., Martello, S. ve Vigo, D. (2002). Recent advances on two-dimensional bin packing problems. *Discrete Applied Mathematics, 123*, 379-396.

Maarouf, W., Barbar, A. ve Owayjan, M. (2008). A new heuristic algorithm for the 3D bin packing problem. *Elleithy K. (eds) Innovations and Advanced Techniques in Systems, Computing Sciences and Software Engineering* (s. 342-345). Beyrut: Springer, Dordrecht.

McCloskey, B. ve Shankar, A. (2005). *Approaches to bin packing with Clique-Graph conflicts.* Berkeley: EECS Department, University of California.

Minyi, Y. (1990). *A simple proof of the inequality FFD(L) <= 11/9 OPT(L) +1, all L for the FFD bin-packing algorithm.* Research Institute for Discreate Mathematics University, Bonn.

Naessens, H., Meyer, H. ve Baets, B. (2002). Algorithms for the computation of T-transitive closures. *IEEE Transactions on Fuzzy Systems*, 541-551.

Nasibov, E. N. (2004). An algorithm for constracting an admissible solution to bin packing problem with fuzzy constrains. *Journal of Computer and Systems Sciences International, 43*, 205–212.

Nasibov, E. N., Şenol, S. ve Nasibova R. (2004). Problem of optimal task allocation with fuzzy competence matrix. *Automatic Control and Computer Sciences*, 23-33.

Nasibov, E. ve Kınay, A. (2006). Kaliteli iş paylaşım problemi için bulanık mantık yaklaşımı. *İstanbul Ticaret Üniversitesi Fen Bilimleri Dergisi*, 13-22.

Nasibov, E., Atilgan, C., Berberler, M. ve Nasibov, R. (2015). Fuzzy joint points based clustering algorithms for large data sets. *Fuzzy Sets and Systems*, 270, 111-126.

Nasibov, E., Vahaplar, A., Demir, M. ve Okur, B. (2016). A fuzzy logic approach to predict the best fitted apparel size in online marketing. *10th International Conference on Application of Information and Communication Technologies*, 599-602.

Pedrycz, W. ve Gomide, F. (2007). *Fuzzy systems engineering toward human-centric computing.* New Jersey: Wiley-IEEE Press.

Rardin, R. (1998). *Optimization in operations research.* Upper Saddle River: Prenice-Hall.

Shaw, P. (2004). A Constraint for Bin Packing. P. Shaw içinde, *Principles and practice of constraint programming – CP 2004* (s. 648-662). Valbonne, France: Springer Berlin Heidelberg.

Simchi-Levi, D. (1994). New worst-case results for the bin-packing problem. *Naval Research Logistics 41*, 579-585.

Stern, K. (2014). *Software and algorithms*. 6 Mart 2017, https://github.com/KevinStern/software-and-algorithms adresinden alındı

Tang, X., Li, Y., Ren, R. ve Cai, W. (2016). On first fit bin packing for online cloud server allocation. *2016 IEEE International Parallel and Distributed Processing Symposium (IPDPS)*.

Vahaplar, A., Demir, M., Okur, B. ve Nasibov, E. (2016). Bulanık mantık yaklaşımı ile uygun giysi bedeni belirleme üzerine bir uygulama. *International Conference on Computer Science and Engineering*, 725-730.

Wallace, M. (2006). Computationally efficient sup-t transitive closure for sparse fuzzy binary relations. *Fuzzy Sets and Systems*, 341-372.

Wu, Y., Li, W., Goh, M. ve Souza, R. D. (2011). Three-dimensional bin packing problem with variable bin height. *European Journal of Operational Research*, 347-355.

Zadeh, L. (1965). Fuzzy sets. *Information and Control*, 338-353.