E. Ü. FEN BİLİMLERİ ENSTİTÜSÜ

# EGE ÜNİVERSİTESİ

## YÜKSEK LİSANS TEZİ

# METASEZGİSEL ALGORİTMALAR İÇİN BİR İŞBİRLİKÇİ SİSTEM VE UYGULAMASI

**Barış Tekin TEZEL**

**Tez Danışmanı : Doç. Dr. Ali Mert**
**İstatistik Anabilim Dalı**

**Bilim Dalı Kodu : 406.02.11**
**Sunuş Tarihi : 30.06.2015**

**Bornova-İZMİR**
**2015**

**EGE ÜNİVERSİTESİ FEN BİLİMLERİ ENSTİTÜSÜ**

**(YÜKSEK LİSANS TEZİ)**

# METASEZGİSEL ALGORİTMALAR İÇİN BİR İŞBİRLİKÇİ SİSTEM VE UYGULAMASI

**Barış Tekin TEZEL**

**Tez Danışmanı: Doç. Dr. Ali MERT**

**İstatistik Anabilim Dalı**

**Bilim Dalı Kodu: 406.02.11**
**Sunuş Tarihi: 30.06.2015**

**Bornova-İZMİR**

**2015**

Barış Tekin TEZEL tarafından yüksek lisans tezi olarak sunulan "Metasezgisel Algoritmalar için Bir İşbirlikçi Sistem ve Uygulaması" başlıklı bu çalışma, E.Ü. Lisansüstü Eğitim ve Öğretim Yönetmeliği ile E.Ü. Fen Bilimleri Enstitüsü Eğitim ve Öğretim Yönergesi'nin ilgili hükümleri uyarınca tarafımızdan değerlendirilerek savunmaya değer bulunmuş ve 30.06.2015 tarihinde yapılan tez savunma sınavında aday oybirliği/oyçokluğu ile başarılı bulunmuştur.

**Jüri Üyeleri:** **İmza**

**Jüri Başkanı : Prof. Dr. Urfat Nuriyev**

**Raportör Üye : Prof. Dr. Efendi Nasiboğlu**

**Üye : Doç. Dr. Ali Mert**

**EGE ÜNİVERSİTESİ FEN BİLİMLERİ ENSTİTÜSÜ**

**ETİK KURALLARA UYGUNLUK BEYANI**

E.Ü. Lisansüstü Eğitim ve Öğretim Yönetmeliğinin ilgili hükümleri uyarınca Yüksek Lisans Tezi olarak sunduğum "Metasezgisel Algoritmalar için Bir İşbirlikçi Sistem ve Uygulaması" başlıklı bu tezin kendi çalışmam olduğunu, sunduğum tüm sonuç, doküman, bilgi ve belgeleri bizzat ve bu tez çalışması kapsamında elde ettiğimi, bu tez çalışmasıyla elde edilmeyen bütün bilgi ve yorumlara atıf yaptığımı ve bunları kaynaklar listesinde usulüne uygun olarak verdiğimi, tez çalışması ve yazımı sırasında patent ve telif haklarını ihlal edici bir davranışımın olmadığını, bu tezin herhangi bir bölümünü bu üniversite veya diğer bir üniversitede başka bir tez çalışması içinde sunmadığımı, bu tezin planlanmasından yazımına kadar bütün safhalarda bilimsel etik kurallarına uygun olarak davrandığımı ve aksinin ortaya çıkması durumunda her türlü yasal sonucu kabul edeceğimi beyan ederim.

30/06/2015

Barış Tekin TEZEL

# ÖZET

## METASEZGİSEL ALGORİTMALAR İÇİN BİR İŞBİRLİKÇİ SİSTEM VE UYGULAMASI

TEZEL, Barış Tekin

Yüksek Lisans Tezi, İstatistik Anabilim Dalı
Tez Danışmanı: Doç. Dr. Ali MERT
Haziran 2015, 68 sayfa

Optimizasyon problemleri, belirli sınırları olan parametrelere bağlı olarak en iyi durumu bulmayı gerektiren fonksiyonlar olarak tanımlanır. Optimizasyon alanında, amaç tüm kısıtları sağlayan pek çok alternatif çözümden en iyi veya yaklaşık en iyi çözümü bulmaktır.

Üst sezgisel, optimizasyon problemlerinin yaklaşık çözümlerini bulmakta son derece efektif bir metotdur. Ancak, üst sezgiseller kullanıldığında algoritma seçim problemi ortaya çıkar. Bu problem, çalışılan optimizasyon probleminin, performans ölçüsünü maksimize edecek şekilde hangi algoritma ile çözüleceğidir.

Bu tezde amaçlanan algoritma, seçim problemiyle başa çıkabilmek için farklı stratejileri zekice birleştiren bir işbirlikçi sistemi kullanmayı önermektir. Farklı üst sezgisellerin zeki kombinasyonunun daha etkili, daha esnek ve daha dayanıklı yaklaşımlar sağlaması beklenmektedir. Fakat bu daha fazla dayanıklılık elde etmek amacıyla daha az hassasiyet gerektirir. Bu durum, esnek hesaplama ile dizayn edilmiş metodoloji ile üretilecektir.

Algoritma seçim problemine ek olarak, algoritma parametrelerinin ayarlanması, iyi sonuçlar elde etmekte anlamlı öneme sahiptir. Bu nedenle, önerilen işbirlikçi sistem, esnek hesaplama tekniklerine dayalı olarak, parametrelere hassas ayarlama yapar.

Bu yaklaşımın işe yararlılığını test etmek amacıyla, önerilen sistem ve bireysel üst sezgisellerden alınan sonuçlar arasında karşılaştırma yapılmıştır.

**Anahtar kelimeler:** Optimizasyon, Üst sezgisel, Esnek Hesaplama , İş Birlikçi Sistem.

# ABSTRACT

## A COOPERATİVE SYSTEM FOR METAHEURISTICS AND ITS APPLICATION

TEZEL, Barış Tekin

Supervisor: Assoc. Prof. Dr. Ali MERT

June 2015, 68 pages

Optimization problems is defined as function required finding optimum state depends on parameters, which are certain limitations. In field of optimization, aim is to find the optimal solution or approximate solution from many alternative solution that provides all the restrictions.

Metaheuristic is an extremely effective method for finding approximate solutions to optimization problems. However, when metaheuristics used, algorithm selection problem arises. This problem is to decide that studied optimization problem is solved by in which algorithm so as to maximize performance measure.

In this thesis, purpose is to use a cooperative system for combine different metaheuristics brilliantly to deal with algorithm selection problem. An intelligent combination of different metaheuristics is expected to provide more flexible, more efficient and more robust approaches. But it requires the less precision in order to obtain more robustness. This situation is generated based on a methodology designed with soft computing.

In addition algorithm selection problem, adjusting algorithm parameters have significant importance to obtain good results. For this reason, proposed cooperative system give a fine tuning of parameters depends on soft computing techniques.

In order to test usefulness of this approach, comparison is made between results obtained by proposed system and individual metaheuristics.

**Key words:** Optimization, Metaheuristic, Soft Computing, Cooperative System.

# TEŞEKKÜR

Bana bu yüksek lisans tezi konusunda çalışma olanağı tanıyan ve tez süresince yardımlarını esirgemeyen tez danışmanım Doç. Dr. Ali Mert'e çok teşekkür ederim. Tez çalışmalarıma bilgi ve yönlendirmeleriyle destek olan Prof. Dr. Urfat Nuriyev, Prof. Dr. Efendi Nasiboğlu, Yrd. Doç. Dr. Ayşe Övgü Kınay ve Yrd. Doç. Dr. Murat Erşan Berberler'e de teşekkürü bir borç bilirim.

Tüm eğitim hayatım boyunca destkelerini hissettiğim aileme, dostlarıma ve hocalarıma ayrıca teşekkür ederim.

# İÇİNDEKİLER

Sayfa

# İÇİNDEKİLER(devam)

# ŞEKİLLER DİZİNİ

Şekil Sayfa

# ŞEKİLLER DİZİNİ(devam)

Şekil Sayfa

# ÇİZELGELER DİZİNİ

# 1.GİRİŞ

Optimizasyon problemleri, bir sistemde var olan belirli sınırlamalar içinde ki parametreler ve bu parametrelere bağlı olarak en iyi halinin bulunulması istenen (minimum veya maksimum) belirli bir amaç fonksiyonu olarak tanımlanmaktadır. Optimizasyon problemlerinde, genellikle amaç tüm sınırlamaları sağlayan pek çok alternatif çözüm (uygun çözümler) arasından optimum çözümü (amaç fonksiyonunun optimum değerini almasını sağlayan) veya yaklaşık çözümü (amaç fonksiyonunun optimum değere yakın değerler almasını sağlayan) bulmaktır. Bu tarz problemler çok yaygın olduğu kadar, gerçekten karmaşık da olabilirler. Bu nedenden ötürü bu tarz problemleri çözmek için pek çok farklı strateji geliştirilmiştir.

Bilgisayar Bilimleri, Matematik, İstatistik vb. alanlarda kullanılan algoritma tiplerinden birisi de sezgisel algoritmalardır. Temel olarak çalışmalarında kesinlik bulunmayan bu algoritmalar ya her zaman aynı performans ile çalışmaz ya da her zaman en iyi sonucu vermeyi garanti etmez ancak yine de problemin her hangi bir çözümünü bulmak için kullanışlı algoritmalardır. Optimizasyon problemlerinin en iyi çözüm stratejilerinden biri de sezgisellerin ileri düzeyi olarak da düşünülen üst sezgisellerdir.

Üst sezgiseller, optimizasyon problemlerinin yaklaşık çözümlerini bulmak için son derece etkin yöntemlerdir. Ancak, üst sezgiseller kullanıldığında algoritma seçim problemi ile karşılaşılır (Rice, 1976). Performans seçim problemi, performans ölçüsünü en büyükleyecek şekilde, ilgilenilen problemin bir durumunda hangi algoritma ile çözüleceğine karar verilmeye çalışılmasıdır. Bu probleme en yaygın çözüm yaklaşımı, belirli bir dizi algoritmanın durum üzerinde denenip, en iyi performans vereni kullanmaktır.

Bu çalışmada amaçlanan algoritma, seçim problemiyle başa çıkabilmek için farklı stratejileri zekice birleştiren bir işbirlikçi sistemi kullanmayı önermektir. Farklı üst sezgisellerin zeki kombinasyonunun daha etkili ve daha esnek yaklaşımlar sağlaması, farklı üst sezgisellerin birleşimi ile daha dayanıklı yaklaşımlar elde edilmesi beklenmektedir. Fakat daha fazla dayanıklılık amaçlı farklı üst sezgisel bileşimleri daha az hassaslık gerektirebilir (Cadenes et al., 2006). Bu gereksinimi sağlamak için kullanılacak metodoloji, Esnek Hesaplama yöntemlerine dayalıdır.

İfade edilen biçimde oluşturulan işbirlikçi sistem, düşük dayanıklılığa sahip yaklaşımlarla çözülmesi çok zor karmaşık problemleri çözmek amacıyla, kendine ait çıkarsama mekanizması ve üst sezgisellerin alan bilgilerini birleştiren bir arama metoduna sahip, yeni bir hesaplama aracı sağlar.

Çalışmada önerilen yöntemi denemek için tek kısıtlı basit bir tam sayılı problem olan 0-1 sırt çantası problemi seçilmiştir. Bunun nedeni 0-1 Sırt Çantası Probleminin hem uygulama kolaylığı hem de karmaşık problemlerde, Bir Boyutlu Kesme Problemi ve Bidon Paketleme Probleminde olduğu gibi, sıklıkla bir alt problem olarak ele alınmasıdır.

Bu tez, üst sezgiseller için bir işbirlikçi sistem tasarlamak üzerinedir. Öncelikle, Bölüm 2'de Sırt Çantası Problemi tanıtılmıştır. Bölüm 3'de ise Sezgisel Algoritmalar incelenmiştir. Bölüm 4'te Bulanık Çıkarsamaya değinilmiştir. Bölüm 5'te önerilen sistemin genel şeması ortaya konmuş, 0-1 Sırt Çantası Problemi için tasarlanan sistem verilmiş ve hesaplama denemeleri sunulmuştur. Sonuç bölümü olan Bölüm 6'da tezin çıktıları değerlendirilmiştir.

## 2.SIRT ÇANTASI PROBLEMİ

Sırt çantası problemi, 1957' de Dantzig'in öncü çalışmasından beri yoğun olarak çalışılan bir kombinatoriyel optimizasyon problemidir (Dantzig, 1957). Optimizasyon problemi olarak ele alındığında çözümü bakımından ünlü bir NP-Zor problemi iken karar verme problemi olarak düşünüldüğünde NP-Tam'dır. Aynı zamanda Karp'in 21 NP-Tam probleminden biridir (Karp, 1972).

Varsayalım ki, bir otostopçu, sırt çantasını kendine maksimum konforu sağlayacak çeşitli mümkün objeler arasından sırt çantası kapasitesini aşmayacak şekilde dolduracak olsun. Bu problemin matematiksel formülasyonu için 1'den $n$ 'e kadar numaralanmış objeler ve aşağıda tanımlı $x_j (j = 1, \dots, n)$ ikili değişkenlerini içiren bir vektör olsun.

$$x_j = \begin{cases} 1, & j.eşya\ seçilirse \\ 0, & aksi\ halde \end{cases}$$

O zaman, $p_j$, $j$'nci objenin sağladığı fayda ve $w_j$ onun boyutu, $c$ ise sırt çantasının kapasitesi olduğunda $x_j$ ikili değişkenlerini içeren herhangi bir vektörün uyması gereken kısıt aşağıdaki gibidir.

$$\sum_{j=1}^{n} w_j x_j \leq c$$

En büyüklenecek amaç fonksiyonu ise;

$$\sum_{j=1}^{n} p_j x_j$$

şeklinde olmalıdır. Genelliği bozmamak adına tüm $w_j$, $p_j$ ve $c$'lerin pozitif olduğu varsayılsın (Martello et al., 1999).

Problem ifade edildikten sonra, problemi anlamaya çalışan kişi tarafından problemin kolayca çözülebileceği düşünülebilir. Akla ilk gelen çözüm yaklaşımı tüm mümkün $x_j$ kombinasyonları içeren $\boldsymbol{X}$ vektörlerini denemek ve en iyisini seçmek olabilir. Böylece vektörlerin sayısı obje sayısına bağlı olarak $2^n$ olacaktır. Bu durumda, (tüm kombinasyonları hesaplamak için) varsayımsal olarak saniyede bir milyar vektör hesaplama kapasitesi olan bir bilgisayarda dahi, n=60 olduğunda

30 yıldan fazla, n=61 olduğunda 60 yıldan fazla ve n=65 olduğunda ise on yüzyıldan fazla zaman gerekmektedir (Martello and Toth, 1990).

### 2.1. 0-1 Sırt Çantası Problemi

0-1 Sırt Çantası Problemi faydayı maksimize ederken objelerin toplam boyutunun sırt çantasının kapasitesini aşmayacak şekilde $n$ adet objenin bir alt kümesinin bulunması problemidir.

Dolayısıyla karar değişkeni, $j$'nci maddenin sırt çantasına dahil edilip edilmeyeceğini 0-1 türünden tanımlar.

$$x_j = \begin{cases} 1, & j.eşya\ seçilirse \\ 0, & aksi\ halde \end{cases}$$

Bu durumda problem aşağıdaki en büyükleme problemi biçiminde yazılabilir.

$$\text{Enb} \sum_{j=1}^{n} p_j x_j$$

Kısıtlar

$$\sum_{j=1}^{n} w_j x_j \leq c$$

$$x_j \in \{0,1\}, j = 1, \dots, n$$

Burada $x_j$ ikili bir değişkendir. $j$'nci obje sırt çantasına dahil edilir ise 1, edilmez ise 0 değeri alır.

### 2.2 Çoktan-Seçmeli Sırt Çantası Problemi

0-1 Sırt Çantası Probleminin farklı bir genelleşmiş halidir. Bu problem, 0-1 Sırt Çantası Probleminde her bir $j$'nci obje $k$ adet $N_i, i = 1, \dots, k$ sınıfının en az birinden kesinlikle seçilecek şekilde faydayı maksimize eden halidir.(Pisinger D., 1995)

Formülasyonu aşağıdaki gibidir.

$$\text{Enb} \sum_{i=1}^{k} \sum_{j \in N_i} p_{ij} x_{ij}$$

Kısıtlar

$$\sum_{i=1}^{k} \sum_{j \in N_i} w_{ij} x_{ij} \leq c$$

$$\sum_{j \in N_i} x_{ij} = 1\,, \quad i = 1, \dots, k$$

$$x_{ij} \in \{0,1\}, i = 1, \dots, k\,, j \in N_i$$

Burada ikili değişken $x_{ij} = 1$ durumunda $j$ 'nci obje $i$ 'nci sınıftan seçilmiştir anlamına gelir. $\sum_{j \in N_i} x_{ij} = 1\,, i = 1, \dots, k$ kısıdı ise her bir sınıftan tam olarak bir eşyanın seçildiğini garanti eder (Nauss, 1978).

### 2.3 Sınırlı Sırt Çantası Problemi

Eğer her bir $j$ obje tipinden $m_j$ miktarına kadar sahip olunursa, bu durumda sınırlı sırt çantası problemi aşağıdaki gibi olacaktır.

$$\text{Enb} \sum_{j=1}^{n} p_j x_j$$

Kısıtlar

$$\sum_{j=1}^{n} w_j x_j \leq c$$

$$x_j \in \{0,1, \dots, m_j\}, j = 1, \dots, n$$

Burada $x_j$'ler amaç fonksiyonu değerini en büyüklemek için her eşya tipinden alınan eşya sayısıdır.

Sınırsız Sırt Çantası Problemi ise Sınırlı Sırt Çantası probleminin özel bir hallidir. Her bir eşya tipinin sayısı limitsizdir. Matematiksel olarak ifade edilecek olursa;

$$\text{Enb} \sum_{j=1}^{n} p_j x_j$$

Kısıtlar

$$\sum_{j=1}^{n} w_j x_j \leq c$$

$$x_j \geq 0 \; Tamsayı, j = 1, \dots, n$$

şeklinde olacaktır. Aslında Sınırsız Sırt Çantası Problemi her bir $x_j$ objesi için sınırın sırt çantası kapasitesi $c$' ye eşit olması durumudur. Ancak, bunu böyle ifade etmek çözüm için bir avantaj sağlamayacağı için genelde böyle ifade edilmez. Problem bu haliyle hala NP-Zor sınıfının bir üyesidir (Lueker, 1975). Fakat n=2 durumu için polinomiyel zamanda çözülebilir (Hirschberg and Wong, 1976; Kannan, 1980).

### 2.4. Altküme Toplamı Problemi

Altküme Toplamı Problemi tanımı; $A = \{a_1, a_2, \dots, a_n\}$, $N$ doğal sayılar kümesinin bir $A \subset N$ altkümesi ve $b \in N$ tamsayısı verilmiş olsun; $\exists S, S \subseteq \text{A} \; \sum_{s \in S} s = b$ yani $A$ kümesinin elemanları toplamı $b$' ye eşit olan bir $S$ altkümesi olup-olmadığı sorulur, şeklinde yapılmıştır (Cormen et al., 2001).

Bu problemi 0-1 Sırt Çantası Problemi biçiminde ifade etmek istersek her bir j'nci objenin faydasi $p_j$ ile boyutu $w_j$ ile eşit olacak biçimde;

$$\text{Enb} \sum_{j=1}^{n} w_j x_j$$

Kısıtlar

$$\sum_{j=1}^{n} w_j x_j \leq c$$

$x_j \in \{0,1\}, j = 1, \ldots, n$

şeklinde formüle edilebilir.

### 2.5 Para Üstü Problemi

Varsayılsın ki, bir kasiyer $c$ değerindeki para üstünü, $w_1, \ldots, w_n$ değerlerine sahip madeni paralardan en az sayıda kullanarak geri vermek durumunda olsun. Bu durumda problem tanımı aşağıdaki gibi olur.

$$\text{Enk} \sum_{j=1}^{n} x_j$$

Kısıtlar

$$\sum_{j=1}^{n} w_j x_j = c$$

$x_j \geq 0 \, , j = 1, \ldots, n$

Bu problem Sınırsız Sırt Çantası Probleminin tüm objeler için fayda değerlerinin 1 olduğu durumda en küçükleme halidir.

### 2.6 Çoklu Sırt Çantası Problemi

Mümkün olan en yüksek miktarda kar elde etmek için N adet eşya, birbirinden farklı $c_i$ kapasitelerde olabilen $m$ adet sırt çantasına yerleştirilmek için seçildiğinde, bu probleme Çoklu Sırt Çantası Problemi denir.

$$\text{Enb} \sum_{i=1}^{m} \sum_{j=1}^{n} p_{ij} x_{ij}$$

Kısıtlar

$$\sum_{j=1}^{n} w_j x_{ij} \leq c_i,\ i = 1, \dots, m$$

$$\sum_{i=1}^{m} x_{ij} \leq 1,\ \ j = 1, \dots, n$$

$$x_{ij} \in \{0,1\},\ \ i = 1, \dots, m, j = 1, \dots, n$$

Burada $x_{ij} = 1$; $j$. eşyanın $i$. sırt çantasına yerleştirildiğini ifade etmektedir. $\sum_{j=1}^{n} w_j x_j \leq c_i$ kısıtı ise her bir sırt çantasının kapasitesinin geçilmediğini güvence altına alır. $\sum_{i=1}^{m} x_{ij} \leq 1$ kısıt ile de bir eşyanın en fazla bir kez seçilmesi sağlanmıştır.

### 2.7 Bidon Paketleme Problemi

Çok kullanışlı bir model olan Bidon Paketleme Probleminde, verilen $n$ eşyanın, eşit kapasiteli bidonlardan mümkün olan en az sayıda kullanılarak paketlenmesi istenmektedir. Böylece model:

$$\text{Enk} \sum_{i=1}^{n} y_j$$

Kısıtlar

$$\sum_{j=1}^{n} w_j x_{ij} \leq c_y,\ i = 1, \dots, n$$

$$\sum_{i=1}^{n} x_{ij} = 1,\ \ j = 1, \dots, n$$

$$y_i \in \{0,1\},\ \ i = 1, \dots, n$$

$$x_{ij} \in \{0,1\},\ \ i = 1, \dots, m, j = 1, \dots, n$$

şeklinde olacaktır. Burada $y_i$ $i$. bidonun kullanılıp kullanılmadığını ifade etmektedir. $x_{ij}$ ise $j$. eşyanın $i$. bidona yerleştirilme durumunu temsil eder.

$\sum_{i=1}^{n} x_{ij} = 1$ kısıtı ile her bir eşyanın paketlendiğinden emin olunur. $\sum_{j=1}^{n} w_j x_{ij} \leq c_y$ kapasite kısıt bidonların kapasitelerinin aşılmamasını sağlar.

# 3.SEZGİSEL ALGORİTMALAR

Optimal çözümün neye benzediği, nasıl çözüme gidileceği tam olarak bilinmediği, hakkında çok az bilgi sahibi olunulan ve uzayın çok büyük olması nedeniyle tüm uzayı aramanın mümkün olmadığı; yani herhangi bir deterministik metotla makul zamanlarda çözülemeyen problemlere zor optimizasyon problemleri denir. Bu problemler sürekli veya kesikli, kısıtlı veya kısıtsız, tek veya çok amaç fonksiyonlu olmak üzere kategorize edilebilir (Boussaid et al. 2013). Bu tip problemlere tatmin edici çözümler bulmak için sezgisel algoritmalar kullanılabilir.

Sezgisel algoritmalar; bilgisayar bilimleri, yapay zekâ ve optimizasyonda kullanılan, arama aşamasını sezgiye dayalı bilgi rehberliğinde gerçekleştiren algoritmalardır. Bu algoritmaların arama performansları doğrudan sezgisel bilginin kalitesine bağlıdır (Russell and Norvig 2010). Bu nedenden ötürü bu algoritmaların çözüm uzayında optimum çözüme yakınsaması ispat edilemez. Her ne kadar bu algoritmaların çalışmasında kesinlik bulunmasa da genelde iyiye yakın çözümleri, makul sürelerde üretirler. Sezgisel arama algoritmalarına örnek olarak:

- A* Araması (A* Search)
- Demet Araması (Beam Search)
- Tırmanış Araması (Hill Climbing)
- En İyi Öncelikli Arama (Best First Search)

Temel de sezgisel algoritmalar, bir problemin çözümünde klasik metotların kesin çözümü bulma konusunda başarısız olduğu veya çok yavaş kaldığı ortamlarda kullanmak için tasarlanmışlardır. Böylece, sezgisel algoritmalar NP sınıfı problemlerin çözümü için kullanışlı bir alternatif oluştururlar.

## 3.1.Sezgisel Algoritmalara Gerek Duyulma Sebepleri

Ele alınan problemin, kesin çözüm bulma işlemi çok fazla süre aldığı veya kesin çözümün tanımlanamadığı bir duruma sahip olunduğunda; karar değişkenlerinde karmaşık karşılıklı bağımlılık durumu olduğunda; doğrusal olmayan amaç fonksiyonu, kısıtlar ve tanımlanabilecek veya kolaylıkla

tanımlanabilecek bir matematiksel fonksiyon olmadığında; teorik optimum çözüme yakınsamak yeterli olduğu durumlarda sezgisel algoritmalara gerek duyulabilir (Eles and Peng, 2010).

### 3.2.Sezgisel Algoritmalar İçin Değerlendirme Kriterleri

Sezgisel algoritmaları değerlendirmek için bir çok kriter vardır. Bunlardan en önemlisi optimalliktir. Yani ele alınan problem için birden fazla çözüm var ise, algoritmanın bu çözümler arasından en iyisini bulup bulamayacağıdır.

Bir diğer değerlendirme kriteri ise bütünlüktür. Bütünlük, problemin tüm çözümlerine algoritmanın ulaşabilme kabiliyetidir. Çoğu sezgisel, problemin yalnızca tek çözümünü ifade etme eğilimindedir.

Göz önünde bulundurulması gereken başka bir kriter ise elde edilen sözde çözümün, gerçek çözüme yakınlık derecesi ve aynı şartlar altında tekrarlanan her bir deneyde aynı sonucu verme derecesi olan doğruluk ve kesinliktir.

Hesaplama zamanı kriteri ise bir sezgisel algoritmanın problemi makul bir zamanda çözmesini ele alır. Sezgisel algoritmalar farklı tip problemlerde cevaba yakınsama hızı bakımından farklılık gösterirler. Özellikle bu kriter hangi problem tipi için hangi sezgisel algoritmanın kullanacağının belirlenmesinde önemli bir işleve sahiptir.

Ayrıca bir sezgisel algoritmayı gerçeklenebilirlik, esneklik, dayanıklılık ve analiz edilebilirlik gibi pek çok kriter altında da incelemek mümkündür. Ancak sezgisel algoritmaların dayandıkları teorik yapı çok ayrıntılı olmadığından, pek çok durumda sezgiselin yeterli olup olmadığını söylemek kolay değildir.

Sezgisel arama algoritmaları; özelikle iteratif gelişme tabanlı olanlar, çoğu zaman başlangıç çözümü veya çözümlerine bağlı olarak yerel optimum çözüm üretirler. Bu yerel optimum çözümler, evrensel optimum çözümden çok uzak olabilir. Bu durumla başa çıkabilmek için farklı başlangıç çözümleri ile yeniden başlatılması veya daha karmaşık komşuluk yapıları oluşturulması gibi yöntemler denenebilir (Karaboğa, 2011). Yine de çözüm kalitesinde iyileşme elde edilemeyebilir. Özellikle son 30 yılda, bu dezavantajı ortadan kaldırmak amacıyla çözüm uzayını etkin bir şekilde aramayı sağlayacak temel sezgisel yöntemleri birleştirmeye çabalayan yeni yaklaşık yöntemler üzerine çalışılmaktadır. Bunlar

temelde üst sezgisel algoritmalar ve hiper sezgisel algoritmalar olarak ele alınabilir.

### 3.3. Üst Sezgiseller

Sezgisel metotların zor optimizasyon problemleri için geniş kapsamlı araçlar oldukları; iyi (göreceli olarak optimuma yakın) çözümler ile makul zaman ve maliyet arasında denge kurdukları deneysel olarak kanıtlanmıştır. Fakat sezgisel algoritmalar çoğu zaman ilgilenilen problemin karakteristiği ele alınarak geliştirilmiş ve dizayn edilmiştir. Bu sakıncayı gidermek adına üst sezgisellerin göz ardı edilemeyecek bir avantajı vardır (Glover, 1977). Bu algoritmalar problemden bağımsız algoritmalardır; yani herhangi özel bir probleme kolaylıkla adapte edilebilir. Gerçekten de isminde geçen "meta" yunan ön ekini alma sebebi tam olarak budur. "Meta" ön eki bu algoritmaların "yüksek seviyeli" sezgiseller olduğunu belirtmektedir.

Üst sezgiseller çoğu zaman "*Gördüğüm zaman bilirim"* problemlere uygulanır (Luke, 2013). Bu tarz problemler esasen; önceden optimal çözümün ve bu çözüme giden ilkesel yolun neye benzediğinin bilinmediği; problem haklında sezgisel bilginin çok az olduğu ve çözümün uzayının büyüklüğü nedeniyle kapsamlı aramanın gerçekleştirilemeyeceği; fakat bir aday çözüm verildiğinde bunun ne kadar iyi bir çözüm olduğunun test edilebileceği; dolayısıyla iyi bir çözüm görüldüğünde anlaşılabilecek problemlerdir.

Örneğin; futbol oynayan bir robotun optimal talimat dizisi aransın. Böyle bir durum için bir simülatör var olsun ve verilecek herhangi bir talimat dizisini test ederek, kalitesini ("*Gördüğüm zaman bilirim")* belirleyebilsin. Sonuç olarak; bir robotun talimat dizisinin genel olarak neye benzediği bilinmesine karşın optimal dizinin tam olarak neye benzediği ve bu diziye nasıl ulaşılacağının bilinmesi mümkün gözükmemektedir (Luke, 2013).

"*Gördüğüm zaman bilirim"* problemleri üst sezgiseller tarafından ters problem olarak ele alınır. Ters problemler bir aday çözümü kullanan test fonksiyonu $f$ ile bu aday çözüme ait değerlendirme (amaç fonksiyonu değeri) üreten problemlerdir. Ancak burada bir değerlendirme alan $f^{-1}$ ters fonksiyonu ile bu değerlendirmeye (amaç fonksiyonu değeri) ait aday çözümü üretmek imkansız veya hemen hemen imkansızdır.

Optimizasyon metotları bu tarz problemlerin üstesinden gelmek için geliştirilmişlerdir. Fakat pek çok klasik optimizasyon tekniği test fonksiyonu üzerinden kuvvetli varsayımlar gerektirdiği için kullanmak zordur veya mümkün değildir. Buna karşın üst sezgisellerin varsayım ihtiyaçları son derece düşüktür. Her şeye rağmen göz ardı edilmemesi gereken bir diğer husus, üst sezgiseller genelde bilinen diğer hiçbir teknik işe yaramadığında kullandığı yani bir çeşit son çare metodu olduğudur. Buradan da üst sezgisellerin her gün yenileri eklenen, son derece büyük ve önem taşıyan bir problem kümesine hitap ettiğidir.

Hemen hemen tüm üst sezgiseller bazı özellikleri taşır. Bunlar; doğadan ilham almak, stokastik bileşenler içermek, amaç fonksiyonunun Hessian matrisi veya Gradyan vektörünü kullanmamak, basit yerel arama tekniklerinden karmaşık öğrenim tekniklerine kadar çok geniş bir yelpazedeki metotları kullanmak, ve ilgilenilen problemi düzgün çözmek için doğru karar verilmesi gereken bir veya birkaç parametreye sahip olmaktır.

Üst sezgisel algoritmalar çeşitlendirme (arama) ve yoğunlaştırma arasında ki dengeyi kurabilirse, verilen bir optimizasyon problemi üzerinde başarı sağlayabilir. Yoğunlaştırma için çözüm uzayında yüksek kaliteli çözüm içerdiği varsayılan belirli bir bölge belirlenmelidir. Birikmiş arama deneyimlerinin umut vadettiği bazı alanların aranmasına yoğunlaşılması açısından yoğunlaştırma önemlidir. Var olan üst sezgiseller açısından temel fark bu dengeyi kuruş mekanizmalarıdır (Birattari et al., 2001). Üst sezgiselleri sınıflandırma pek çok farklı özelliğe göre çeşitlilik göstermektedir. Bunlardan bazıları; arama stratejileri, bellek kullanımları, komşuluk ilişkileri ve boyuttur. Boyuta göre üst sezgiseller tek çözüm tabanlı (yörünge tabanlı) ve popülasyon tabanlı olmak üzere ikiye ayrılır. Literatürde kullanılan en temel ayırma biçimi budur (Blum and Roli, 2003).

### 3.3.1. Tek Çözüm Tabanlı Üst Sezgiseller (Yörünge Tabanlı)

Tek çözüm tabanlı üst sezgiseller, tek bir başlangıç çözüm ile başlayarak, bir yörünge doğrultusunda arama yapma prensibine göre çalışırlar. Bazıları yerel arama algoritmalarının "zeki" genişlemeleri olarak görülürler. Yörünge tabanlı metotların başlıcaları; Tavlama Benzetimi, Tabu Araması, GRASP (Açgözlü Rassal Adaptif Arama Prosedürü), Değişken Komşuluk Araması olarak ele alınabilir.

### 3.3.1.1. Tavlama Benzetimi

Tavlama Benzetimi metodunun (SA) temeli istatiksel mekaniğe dayanmaktadır (Metropolis et al., 1953). İlk defa S. Kirkpatrick ve arkadaşları tarafından 1983 yılında ortaya atılmıştır (Kirkpatrick et al., 1983). Tavlama Benzetimi maden bilimciler tarafından kullanılan tavlama tekniğinden esinlenilmiştir. Bu teknik bir materyalin yüksek sıcaklığa getirilerek, bu sıcaklıktan yavaş yavaş azaltılması seklinde kısaca tanımlanabilir. Burada sıcaklığın azaltılması işleminin düzgün yapılması durumunda materyale ait çok düzenli minimum enerjili bir katı hal yani süper kafes hali elde edilir.

SA algoritması tavlama sürecini, bir optimizasyon probleminin çözüm sürecine aktarır. Burada problemin minimize edilecek amaç fonksiyonu, materyalin enerjisine benzetilirken, hayali bir $T$ sıcaklığı ise algoritmanın temel kontrol parametresi olarak ele alınabilir.

Algoritma rasgele veya bir sezgisel yardımıyla oluşturulmuş başlangıç çözümün oluşturulması ve $T$ sıcaklık parametresinin belirlenmesi ile başlar. Her bir iterasyonda ise var olan $s$ çözümünün $N(s)$ komşularından rasgele olarak bir $s'$ çözümü seçilir. $s'$ çözümü $T$ sıcaklık parametreli, $s$ ve $s'$ ait olan $f(s)$ ve $f(s')$ amaç fonksiyonlarına bağlı olarak yeni çözüm olarak belirlenir. Burada, eğer $f(s') \leq f(s)$ ise $s'$ kabul edilir ve $s$ ile yer değiştirilir. Diğer taraftan, eğer $f(s') > f(s)$ ise $s'$, $p\big(T, f(s'), f(s)\big) = \exp(-\frac{f(s')-f(s)}{T})$ olasılığı ile kabul edilebilir. $T$ sıcaklığı arama süreci boyunca azalır. Böylece aramanın başlangıç safhalarında kötüye giden hamle kabulü olasılığı yüksekken, zamanla azalacaktır. SA algoritması aşağıdaki gibidir.

**Algoritma 2.1.** ( Tavlama Benzetimi)

**1.** Rasgele olarak başlangıç $s$ çözümü belirle
**2.** $T$ başlangıç sıcaklığını belirle
**3.** **while** durdurma kriteri sağlamadıkça **do**
**4.** **repeat**
**5.** $s' \in \mathrm{N}(s)$ çözümünü rasgele seç
**6.** **if** $f(s') \leq f(s)$ then
**7.** $s \leftarrow s'$;
**8.** **else**
**9.** $p\big(T, f(s'), f(s)\big) = \exp(-\frac{f(s')-f(s)}{T})$ olasılığı ile $s \leftarrow s'$

**10.** **end**
**11.** **unti**l sistemin termodinamik denge 'sine ulaşılana
**12.** $T$ ' yi düşür
**13. end**
**14. return** elde edilen en iyi çözümü

Burada dikkat edilmesi gereken husus, algoritma arama boyunca daha iyi bir çözüme rastlamış olsa bile, başka bir çözüme yakınsayabilir. Bu nedenle arama süresince karşılaşılan en iyi çözümü saklamak SA üzerinde yapılabilecek en temel iyileştirme olacaktır.

SA her ne kadar pek çok sürekli veya kesikli optimizasyon problemine başarı ile uygulanabilse de, bazı kombinatorik problemler için fazla aç gözlü veya güçsüz bulunabilmektedir (Boussaid et al., 2013).

### 3.3.1.2. Tabu Araması

Tabu Araması (TS) 1986 yıllında Glover tarafından geliştirilmiştir (Glover, 1986). TS' nin en belirgin özelliği arama geçmişi kullanması olarak ele alınabilir. Bunu yapmasında ki amaç hem yerel optimumlardan kaçınmak hem de yerel optimumları, global optimuma erişebilmek için bir kılavuz olarak kullanmaktır. Aslında bu ana karakteristiğini, insanların hafıza mekanizmasından esinlenilmesi temeline dayandırır. Bu açıdan bakıldığında hafıza kullanmayan ve geçmişten öğrenim yapmayan SA algoritmasına tam olarak zıt bir arama stratejisi olarak ele alınabilir. Yani TS'da rassallık, SA'da olduğu gibi, vurgulanmaz ve oldukça kısıtlı olarak kullanılır. Arama sürecini çok daha sistematik bir fikre dayandırdığı için kısmen deterministtik bir metot olarak kabul edilir.

TS algoritması, arama sırasında arama uzayı boyuncaki yörüngenin belirli özelliklerini hatırlamak için farklı çeşitlerde hafıza yapıları kullanılır. Tabu listesi en son karşılaşılan çözümleri veya çözümlerin belirli özelliklerini kaydeden ve bu çözümlerin veya belirli özellikleri içeren çözümlerin, listede oldukları süre boyunca, kullanılmasını engelleyen, kısa dönemli hafıza olarak ele alınabilir. Son dönemde ziyaret edilen çözümlere tekrar geri dönülmesini ve daha kötü çözümlerin kabulünü zorlamak pahasına arama uzayında sonsuz döngüye girilmesini engeller. Temel TS algoritması aşağıda verilmiştir.

**Algoritma 2.2.** ( Tabu Araması)

**1.** Rasgele olarak başlangıç $s$ çözümü belirle

2. $Tabulist \leftarrow \emptyset$
3. **while** durdurma kriteri sağlamadıkça **do**
4. En iyi $s' \in N(s) \backslash$Tabulist çözümünü seç
5. $s \leftarrow s'$;
6. $Tabulist$'i güncelle
7. **end**
8. **return** elde edilen en iyi çözümü

Tabu listesinin uzunluğu, arama uzayının hafızasını kontrol eder. Burada uzunluğun kısa olması, aramanın dar bir alana yoğunlaşmasını, uzun olması ise yasaklanan çözüm sayısının fazlalığından ötürü, tam tersi bir etki yaparak çok daha geniş alanlarda aramanın gerçekleşmişini sağlar. Daha dayanıklı bir TS algoritmasına ulaşmak için tabu listesi uzunluğu arama sırasında değiştirebilir (Battiti and Tecchiolli, 1994).

Ek orta dönemli hafıza yapısı ile umut vaat eden alanlarda aramayı yoğunlaştırma eğilimi hem de ek uzun dönemli hafıza yapısı ile de aramayı daha geniş alanlarda çeşitlendirme teşviki gerçekleştirilebilir.

Aspirasyon kriteri olarak bilinen ek orta dönemli hafıza yapısı arama süreci üzerinde büyük ölçekli bir gelişim yaratabilir. Ortada bir döngü riski veya arama sürecine genel durgunluk getirecek bir durum olmamasına karşın tabu listesi cazip bir hareketi engelleyebilir. Böyle durumlarda bir dizi kuraldan oluşan aspirasyon kriteri sağlanıyorsa, tabu listesinin kısıtlamaları geçersiz kılınarak bu hamleye izin verilir.

En sık kullanılan uzun dönemli hafıza yapısı ise frekans tabanlı hafıza yapısıdır. Bu tip hafıza, arama yörüngesi boyunca karşılaşılan çözümlerde belirli bir özelliğin hangi sıklıkta olduğunun kaydını tutar. Böylece çok sık rastlanılan özelliklere sahip çözümlerin ziyaretini engellerken, bu özellikleri taşımayan çözümlerin ziyaretine izin verir.

TS algoritması baskın olarak kombinatorik problemlere uygulanması için tasarlanmıştır. Fakat sürekli optimizasyon problemlerine uygulanmış halleri de mevcuttur (Glover, 1994; Cvijovic and Klinowski 1995; Chelouah and Siarry, 2000).

### 3.3.1.3. Değişken Komşuluk Araması

Değişken Komşuluk Araması (VNS), Hansen ve Mladenovic tarafından ortaya atılmış bir üst sezgiseldir (Mladenovic, 1995; Mladenovic and Hansen, 1997). VNS algoritmasının temel stratejisi verilen bir çözüm üzerindeki komşuluk yapısını dinamik olarak değiştirerek arama yapmaktır. Başlangıç adımı olarak; komşuluk yapılarının bir dizisi tanımlanmalıdır. Komşuluklar keyfi olarak belirlenebileceği gibi, genelde, $N_1, N_2, \dots, N_{max}$ şeklinde artan kardinalitede belirlenir. Esasında komşuluklar arası ilişkinin $N_1 \in N_2 \in \cdots \in N_{max}$ şeklinde olması beklenir. Fakat bazı aramalarda geri dönülmesi mümkün olan çok sayıda çözüm olduğu için bu biçimdeki komşuluk yapıları çok efektif olmayabilir (Blum and Roli, 2003). Komşuluk yapısı oluşturulduktan sonra başlangıç çözümü oluşturulur ve VNS' nin ana döngüsü başlar. Döngü üç ana adımı içerir; bunlar sırası ile; *sallama, yerel arama* ve *hareket* 'dir. Sallama adımında, o anki çözüm $s$' nin *n.* komşuluğundan rasgele olarak bir $s'$ çözümü seçilir. Daha sonra $s'$ çözümü yerel arama için başlangıç çözümü olarak kabul edilerek $s''$ çözümü elde edilir. Yerel arama sonucunda oluşan $s''$ çözümü $s$ çözümümden daha iyisi ise $s''$ çözümü $s$ çözümü yerine geçer ve $n = 1$ olarak atanarak döngü baştan başlatırlar. Aksi durumda bir sonraki komşuluğa geçilerek; bu komşuluk için yeni bir sallama aşaması ile devam edilir. VNS algoritması aşağıda ifade edildiği gibidir.

**Algoritma 2.3.** ( Değişken Komşuluk Araması)

**1.** $N_n, n = 1, \dots, n_{max}$ komşuluk yapıları dizisini seç
**2.** Rasgele olarak başlangıç $s$ çözümü belirle
**3.** **while** durdurma kriteri sağlamadıkça **do**
**4.** $n \leftarrow 1$;
**5.** **while** $n < n_{max}$ **do**
**6.** Sallama: $s$ 'in $n$. komşuluğu $N_n(s)$ 'den rasgele bir $s'$ seç
**7.** $s''$ 'a ulaşmak için $s'$ ile başlatarak yerel arama uygula
**8.** **if** $s''$, $s$ ' den iyi ise **then**
**9.** $s \leftarrow s'$;
**10.** $n \leftarrow 1$;
**11.** **else**
**12.** $n \leftarrow n + 1$;
**13.** **end**
**14. end**
**15. return** elde edilen en iyi çözümü

Seçilen komşuluk yapıların tamamlayıcı olması durumunda yani bir komşuluk yapısı için yerel optimum olan durum bir diğer komşuluk yapısı için olmadığında; VNS algoritmasının etkili olduğu söylenebilir.

### 3.3.2. Popülasyon Tabanlı Üst Sezgiseller

Popülasyon tabanlı üst sezgiseller tek bir çözüm yerine, bir çözümler dizisini (popülasyon) ele alarak arama yaparlar. En çok çalışılan popülasyon tabanlı üst sezgiseller, Evrimsel Hesaplama (EC) ve Sürü Zekası (SI) ile ilişkilidir. EC algoritmaları Darwin'in evrim teorisinden esinlenerek ortaya konulmuşlardır. SI algoritmaları, merkezi bir koordinasyon içermeyen, bireysel bilişsel yetenekler yerine sosyal etkileşim ve kolektif davranışları tercih eden, genelde biyolojik canlılardan esinlenerek arama gerçekleştiren algoritmalardır.

#### 3.3.2.1. Genetik Algoritma

EC, Darwin'in doğada yaşayan canlıların ortamlarına uyum sağlama yeteneklerinden esinlenilen pek çok optimizasyon algoritması için kullanılan genel terimdir. Genetik Algoritma (GA) en çok bilinilen ve en yaygın kullanılan EC tekniği olarak kabul edilebilir. GA ilk olarak, 1970'lerin başında Michigan Üniversitesinde John Holland ve öğrencileri tarafından ortaya konmuştur (Holland, 1975). Temel GA algoritması çok genel ve probleme göre uygulanabilecek pek çok farklı yönü olan bir optimizasyon tekniğidir. Örneğin; çözümün temsili (kromozom), seçim stratejisi, çaprazlama operatörü ve mutasyon operatörü gibi. Kromozomlarda kullanılan en genel temsil tipi sabit uzunluklu ikili dizidir. Basit bit işleme operatörleri, çaprazlama ve mutasyon operatörlerinde istenilen işlemin gerçekleşmesine olanak sağlar. Bu operatörler GA' nın problem çözme stratejisi aşamasının en temel parçalarıdır. Çaprazlama operatörü GA' nın çeşitleme operatörü olarak vurgulanır. Çaprazlama, bireylerin (genelde iki) belirli parçaları değiştirilerek birleştirilmesi esasına dayanır. Bunu gerçekleştirmenin pek çok farklı yolu vardır. Tekdüze veya n noktalı çaprazlama bunlarından sadece iki tanesidir. Algoritmanın çaprazlama oranı parametresi, her bir bireyin çaprazlamaya dâhil olup olmayacağının olasılığını belirtir. Genelde 0,6 ila 1,0 arasında bir değer alır (Bäck and Schwefel, 1993). Yeni nesli üretmek için kullanılacak bireyler, seçim havuzundaki her bir birey için uygunluk değerleri hesaplandıktan sonra, seçim şeması ile belirlenir. Popüler seçim şemaları; Rulet Tekerliği, Turnuva Seçim ve Sıralama Seçim vb. şemalardır (Blickle and Thiele,

1995; Goldberg and Deb, 1991). Çaprazlamadan sonra bireyler mutasyona tabi tutulurlar. Mutasyon aramanın yerel optimumlara takılmaması için arama sürecine bir miktar rassallık katar. Genelde, mutasyon olasılığı populasyonun yüzde 1'den aşağıda tutulur ama yine de problem özelinde değişiklikler gösterebilir. GA' nın temel algoritması aşağıda verildiği gibidir.

**Algoritma 2.4.** ( Genetik Algortima)

1. Rasgele olarak başlangıç popülasyonunu belirle
2. Popülasyon için uygunluk değerlerini hesapla
3. **while** durdurma kriteri sağlanmadıkça **do**
4. Ebeveynleri seç
5. Her bir ebeveyn çift için çaprazlama yap
6. Oluşan yeni nesile mutasyon işlemi uygula
7. Oluşan yeni popülasyon için uygunluk değerlerini hesapla
8. **end**
9. **return** elde edilen en iyi çözümü

GA robotik, yazılım mühendisliği, optik, görüntü işleme ve benzeri pek çok alanda, çok değişik versiyonlar hallinde kullanılmış ve kullanılmaya devam etmektedir.

### 3.3.2.2. Karınca Kolonisi Optimizasyonu

Karınca Kolonisi Optimizasyonu (ACO) zor kombinatoriyal optimizasyon problemleri için Marco Dorigo tarafından önerilen ve doğadan ilham alan bir üst sezgisel algoritmadır. ACO karıncaların yiyecek arama davranışlarını bir problemin çözümü gibi ele alır (Dorigo, 1992; Dorigo and Maniezzo, 1991; Dorigo and Maniezzo, 1996). Karıncalar yuvaları çevrelerinde rasgele olarak gezinerek yiyecek aramaya başlarlar. Buldukları yiyecek kaynakları ile yuva arasında yolları boyunca feromon bırakarak, kendilerinden sonra aramaya çıkacak karıncalar için yol gösterici izler bırakmış olurlar (Dorigo and Blum, 2005). Belirli bir zaman sonunda yiyecek ve yuva arasında ki en kısa yol boyunca feromon miktarı yoğunlaşacaktır ve böylece çok daha fazla karıncaya çekici gelmeye başlayacaktır. ACO bir optimizasyon probleminin çözümlerini oluşturmada ve bir iletişim şemasına göre kalite bilgilerinin birbirlerine aktarılmasında gerçek karıncaların bu karakteristiğinden faydalanır (Dorigo et al. 2006).

$P = (S, \varphi, f)$ şeklinde bir optimizasyon problemi belirtilsin. Burada $S$, $X_i$ $(i = 1, \dots, n)$ sonlu sayıda karar değişkenince tanımlanmış arama uzayı, $\varphi$ değişkenler üzerindeki kısıtlar ve $f(.): S \to R^+$ optimize edilecek amaç fonksiyonu olsun. $s \in S$ problemin tüm kısıtlarını sağlayan herhangi bir uygun çözüm olsun.

ACO verilen kombinatoriyal optimizasyon problemini tam bağlantılı, tepeleri çözümünün bileşenleri ve ayrıtları da bileşenler arası bağlantılar olacak şekilde bir yapı grafı $G_c(V, E)$ olarak kodlar. Verilen herhangi bir çözüm graf üzerindeki uygun bir yürüyüş olarak ele alınır. Burada çözümün her bir bileşeni $c_i^j$ , $v_i^j$ değerli $X_i$ değişkenini ifade eder. Çözüm bileşeni tanımları ele alınan problem özelinde değişiklik gösterebilir. ACO için ele alınan en popüler problem Gezgin Satıcı Problemi (TSP)' dir (Lawler et al., 1985). Karıncaların uygun yürüyüşlerin uygun olan bileşenlerinin kombinasyonuna ihtiyacı vardır. Her çözüm bileşeni $c_i^j$ 'ye bir $\tau_i^j$ feromon miktarı eşlik eder. Bu değer bir çeşit hafıza biçimi olarak da ele alınabilir. Burada feromon miktarları belirli zaman aralığında eşlik ettiği çözüm bileşeninin çekicilik miktarını ifade eder. Tüm çözüm bileşenlerini $C$ ve tüm feromon iz parametrelerini de $\tau$ ile ifade edilsin.

Temel ACO algoritması aşağıdaki gibidir. Parametreler ve feromon izleri oluşturulduktan sonra ACO algoritması 3 aşamayı tekrarlar: Karınca çözümler oluşturma, Artalan hareketler (isteğe bağlı), Feromon güncelleme.

**Algoritma 2.5.** ( Karınca Kolonisi Optimizasyonu)

1. Feromon miktarlarını belirle
2. **while** durdurma kriteri sağlanmadıkça **do**
3. Karınca çözümleri oluşturma
4. Feromon güncelleme
5. Artalan hareketler
6. **end**
7. **return** elde edilen en iyi çözümü

Algoritmanın başlangıç parametreleri belirlenir ve tüm feromon değişkenleri parametre olarak verilen $\tau_0$ değerine eşitlenir.

Karınca çözümleri oluşturma aşamasında; belirli sayıdaki yapay karınca, aşamalı ve stokastik olarak başlangıç çözümü olan, boş kısmi çözümünden $S_p = \emptyset$ başlayarak, ele alınan problem için çözümler üretir. Üretilecek çözümler, oluşturma aşamasının her bir adımının boş çözüme bir uygun çözüm bileşen

eklemesi ile genişletilir. Seçilecek olan bileşen feromon miktarı göz önünde bulundurularak yapılacak olasılıksal bir yöntem ile belirlenir (Dorigo and Maniezzo, 1996).

Artalan hareketler; tek bir karınca ile uygulanamayan herhangi bir merkeze toplama işlemini ifade eder. Oluşan çözümleri kullanan bir yerel arama tekniği örnek gösterilebilir.

Feromon güncelleme; iyi çözümlere ait çözüm bileşenlerini, daha sonraki iterasyonlarda karıncalar için daha cazip hale getirmeyi amaçlamaktadır. Bunun için çalışan iki mekanizması vardır. Bunlar feromon buharlaştırma ve feromon biriktirme mekanizmalarıdır. Feromon buharlaştırma, çözüm oluşturma süreci esnasında, karıncaların takip edeceği izlerin üzerindeki feromon miktarını zamana bağlı olarak azaltma işlemidir. Burada ki amaç; herhangi bir çözüme eklenecek bir çözüm bileşeninin çözüme eklendiğinde ardından gelecek olan karıncalar için cazibesini kaybetmesidir. Böylece arama işleminin belirli bir bölgeye saplanıp kalması engellenmiş olur. Diğer bir mekanizma olan feromon depolama ise tüm karınca çözümleri oluşturulduktan sonra gerçekleştirilir. Yüksek kaliteye sahip çözümlerin bileşenlerinin feromon miktarı artırılır. Böylece bir sonraki iterasyondaki karıncalar için bu bileşenler daha cazip hale gelmesi sağlanır.

ACO algoritması özellikle gerçek hayat problemlerine uygulanması konusunda diğer algoritmalara nazaran daha başarılı olduğu görülür. Ancak çok çeşitli problemlere uyarlandığı da görülmektedir. (Dorigo and Thomas, 2010). Bu problemlerden bazıları; çizelgeme problemleri, araç rotalama problemleri, atama problemleri, görüntü işleme, sınıflama, veri madenciliği gibi sıralanabilir.

#### 3.3.2.3. Yapay Arı Kolonisi Algoritması

Yiyecek arama davranışlarından esinlenen optimizasyon tekniklerinin arasında arı davranışlarını ele alanlar da vardır. Bunlar dinamik ortamlarda arıların bilinen yiyecek kaynaklarını kullanma ve potansiyel olarak daha iyi yeni yiyecek kaynaklarını arama davranışlarını modellemeye çalışır. Bunlardan bazıları Arı Kolonisi Optimizasyonu Üst Sezgiseli (BCO), Yapay Arı Kolonisi Algoritması (ABC) ve Sanal Arı Algoritmasıdır (VBA) (Lucic and Teodorovic, 2003; Karaboga, 2005; Yang, 2005).

ABC algoritmasında, koloni üç gruba bölünmüştür. Bunlar; görevli arı, gözcü arı ve kaşif arıdır. Koloninin yarısı görevli arı iken diğer yarısı gözcü arıdır. Görevli arıların sayısı, kovan etrafında yiyecek kaynaklarının sayısına eşittir. Bir görevli arının ilgilendiği yiyecek kaynağı tükendiği zaman bu arı kaşif arıya dönüşerek rasgele olarak yeni bir yiyecek kaynağı arar. Bir görevli arı ilgilendiği yiyecek kaynağı ile ilgili kovana bilgi taşır ve bunu belirli bir olasılık değeri altında arı dansı ile kovandaki gözcü arılar ile paylaşır. Böylece kovandaki gözcü arılar belirli bir olasılık altında kendilerine bir yiyecek kaynağı seçerler. Bir yiyecek kaynağında nektarın miktarının artması o kaynağın gözcü arılar tarafından seçilme olasılığına da artırır. ABC algoritmasının temel adımları aşağıdaki gibidir.

**Algoritma 2.6.** ( Yapay Arı Kolonisi Algoritması)

1. Başlangıç popülasyonunu belirle
2. **while** durdurma kriteri sağlanmadıkça **do**
3. Görevli arıları yiyecek kaynaklarına yerleştir ve nektarın miktarlarını belirle
4. Kaynakların seçilme olasılıklarını nektarın miktarına göre hesapla
5. Kaynakların seçilme olasılıklarına göre gözcü arıları kaynaklara yerleştir
6. Tükenen kaynakları bırak
7. Kaşif arıları yeni kaynaklar bulması için gönder
8. Şimdiye kadar bulunan en iyi kaynağı hafıza al
9. **end**
10. **return** elde edilen en iyi çözümü

### 3.4. Hiper Sezgisel

Hiper sezgisel algoritmalar ilk olarak 1997 yılında Jörg Denzinger, Matthias Fuchs ve Marc Fuchs tarafından ortaya atılmıştır. Birden fazla yapay zeka tekniğinin seçilim ve birleştirme protokollerini ifade etmek için kullanmışlardır (Jörg Denzinger et al., 1997). Ancak bir makalede ilk defa yayınlanması 2003'de olmuştur (Burke et al., 2003). Buna karşın fikrinin temelleri 1960'ların başlarına kadar dayanmaktır (Fisher and Thompson, 1961; Fisher and Thompson, 1963).

Terim ve anlam olarak birbirine çok yakın olan üst sezgisel algoritmalar ve hiper sezgisel algoritmalar arasındaki ayrım aslında bir probleme çözüm aranılan uzaya göre farklılaşmaktadır. Yani iki yaklaşımda da bir probleme sezgisel olarak

çözüm aranmakta fakat iki yaklaşımdaki çözüm aranan uzaylar farklı olmaktadır. Basitçe üst sezgisel algoritmalarda arama işlemi problemin çözüm uzayında yapılırken hiper sezgisel algoritmalarda arama işlemi sezgisel uzayda yapılır.

Hiper sezgiseller iki temel kategoride değerlendirilir.

- Sezgisel seçen sezgiseller
- Sezgisel üreten sezgiseller

Sezgisel seçen sezgiseller, hedef problemi en iyi şekilde çözecek sezgisel veya sezgisel dizisini belirlemeye çalışırken; sezgisel üreten sezgiseller, var olan sezgisel parçalarını kullanarak hedef soruyu en iyi çözecek sezgiselleri üretmektir.

Bu temel sınıflama dışında, üzerinde çalışılan sezgisellere göre, yapıcı ve karıştırıcı arama olarak da sınıflandırılabilir. Ayrıca, öğrenme süreci açısından, çevrim içi öğrenme ve çevirim dışı öğrenme olarak da sınıflandırılır.

## 4. BULANIK ÇIKARSAMA

Bulanık çıkarsama, bulanık kümeler teorisine dayalı ve aynı zamanda graf teori, topoloji ve optimizasyon gibi yapay zeka, bilgi işleme ve mantıktan, teorik ve uygulamalı matematiğe kadar teorileri kapsayan bir alandır. Bulanık kümeler teorisi 1965'de Zadeh tarafından ortaya konmuştur. Zadeh, burada bulanık kümelerin bazı temel özellikleri ve içermeleri için gerekli ön adımları keşfetme niyetini belirtmiştir (Zadeh, 1965).

Klasik küme teorisi, üye veya üye olmayan bireyler kümesi temel kavramı üzerine inşa edilmiştir. Burada, iyi tanımlanmış bir küme için üye olma ve olmama arasındaki ayrım keskin, net ve kesindir. Herhangi bir elemanın kümeye üye olması için gerekli sınır ve koşullar son derece nettir. Başka bir değişle, "Bu eleman kümeye dâhil mi?" sorusunun cevabı "Evet" veya "Hayır" olarak verilir. Bu durum hem deterministik hem de stokastik durumlarda geçerlidir. Her ne kadar olasılık bir belirsizlik durumu içerse de, sorulacak "Bu elemanın kümeye dâhil olma olasılığı nedir?" sorusunun cevabın "Bu elemanın kümeye dâhil olması olasılığı %90'dır" olsa da sonuçta cevap hala "dâhildir" veya "dâhil değildir" ifadelerinden birisini içermektedir. Kümeye dâhil olma tahmininin doğru olması olasılığının %90 olması, elemanın kümeye %90 üye olduğu ve %10 üye olmadığı anlamına gelmez. Yani, klasik küme teorisinde, bir elemanın kümeye aynı anda üye olması ve olmaması gibi bir duruma izin verilmez. Böylece, pek çok gerçek hayat problemi, içerdiği elemanların sadece tek bir aidiyetliğinin bulunduğu klasik küme teorisi ile açıklanamaz ve ele alınamaz. Bunun tam tersine, bulanık küme teorisi kısmi üyelik durumlarına izin verir ve bir anlamda klasik küme teorisinin bir dereceye kadar genişletilmiş halidir. Bir bulanık kümenin elemanları, aynı zamanda, aynı uzaydaki başka bir kümenin de elemanı olabilirler.

Son yıllarda, bulanık kümeler teorisi, karmaşık sistemler teorisi ve karar süreçlerinde çok daha kullanılır oldu. 1990'dan bu zamana kadar ise bulanık mantık uygulamaları finans, pazarlama ve diğer karar verme problemlerinden mikro kontrol tabanlı sistemler ve geniş ölçekli süreç kontrol sistemlerine kadar pek çok alanda kullanımı yaygınlaşmıştır (Sugeno and Yasukawa, 1993; Karr and Gentry, 1993; Lee, 1990). Doğrusallık içermeyen ve güvenilir olmayan analitik modeller içeren sistemler için bulanık mantık en umut vaat eden yaklaşımlardan biridir aynı zamanda bulanık çıkarsamanın da insan düşünce biçiminin simülasyonunda bir adım olduğu aşikârdır.

Bulanık mantık tekniklerinin veya bulanık mantık tabanlı tekniklerin, karmaşık, doğrusal olmayan veya kötü tanımlanmış problemler üzerinde geleneksel yaklaşımlara karşı ana avantajı nitel bilgi, sistem davranışları üzerindeki uzmanlık ve dinamizmi birleştirebilme yeteneğidir. Bu birleşim özellikle kesin bir matematiksel model ile düzgün açıklanamayan sistemler için bir zorunluluk halindedir. Ancak böyle bir kompozisyonu bulanık yaklaşımlar dışında da elde etmenin mümkün olduğu da göz ardı edilmemelidir.

## 4.1.Bulanık Küme Teorisi ile İlgili Temel Kavram ve Tanımlamalar

### 4.1.1. Bulanık Küme ve Üyelik Notasyonu

Klasik bir $A$ kümesi elemanlar veya objelerin bir araya toplanması olarak tanımlanır. Burada herhangi bir eleman veya obje olan $x$ bir kümeye aittir veya ait değildir şeklinde ifade edilir ve $A$‘ daki $x$’ e ait üyelik fonksiyonu $\mu_A(x)$ aşağıdaki gibi ifade edilir.

$$\mu_A: X \rightarrow \{0,1\}$$

Burada görüldüğü üzere $x$’in $A$’ daki aidiyet değerini temsilen 1 veya 0 değeri alabilir. Bundan dolayı da, bu kümenin tümleyeni ile kesişimi aşağıda ki gibidir.

$$A \cap \bar{A} = \emptyset$$

Diğer yandan bulanık mantık, bulanık kümelere dayalı bir mantıktır. Burada, objelerin veya elemanların bir kümesi, 0 ve 1 kesin değerleri yerine $[0,1]$ aralığında ki doğruluk değeri ile karakterize edilir. Bir bulanık kümenin ilgilendiği uzaydaki elemanların her birini $[0,1]$ aralığına taşıyan fonksiyon, üyelik fonksiyonu olarak ifade edilir.

### 4.1.2. Üyelik Fonksiyonu

Bir $A$ bulanık kümesinin ilgilenilen uzayı $X$ olsun. $A$’ ya ait üyelik fonksiyonu ise;

$$\mu_A: X \rightarrow [0,1]$$

şeklinde tanımlı olsun. Böylece A;

$$A = \{(x, \mu_A(x))\}$$

çifti ile tanımlanır.

Genelde, en çok kullanılan üyelik fonksiyonları, üçgen üyelik fonksiyonu, yamuk üyelik fonksiyonu, sigmoid üyelik fonksiyonu olarak ele alınabilir (Dubois and Prade, 1980; Zimmermann, 1996).

Üçgen üyelik fonksiyonu, (şekil 4.1(a));

$$Tri(x; a, b, c) = \begin{cases} 0 & x < a \\ \dfrac{x-a}{b-a} & a \le x \le b \\ \dfrac{c-x}{c-b} & b \le x < c \\ 0 & x \ge c \end{cases}$$

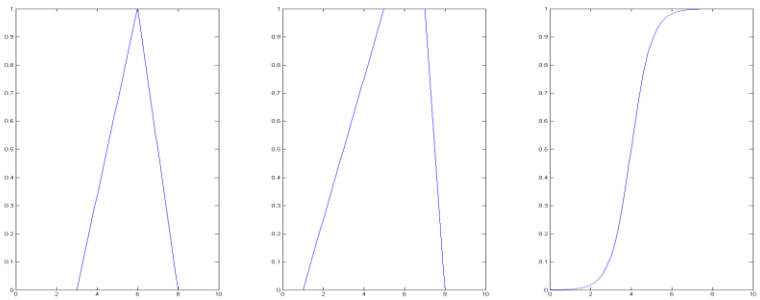
Yamuk üyelik fonksiyonu, (şekil 4.1(b));

$$Tra(x; a, b, c, d) = \begin{cases} 0 & x < a \\ \dfrac{x-a}{b-a} & a \le x \le b \\ 1 & b \le x \le c \\ \dfrac{d-x}{d-c} & c \le x < d \\ 0 & x \ge d \end{cases}$$

Sigmoid üyelik fonksiyonu, (şekil 4.1(c));

$$S(x; a, b) = \frac{1}{1 + e^{-a(x-b)}}$$

şeklinde tanımlanmıştır.



**Şekil 4.1.**(a)Üçgen, (b)Yamuk ve (c)Sigmoid üyelik fonksiyonu

### 4.1.3. Bulanık Küme İşlemleri

Bulanık bir sistemin tasarlanması için bulanık kümeler teorisinde ki işlemleri bilmek ve anlamak çok önemlidir. Bulanık küme işlemleri, kümelerin üyelik fonksiyonlarına dayalı olarak tanımlanır.

Bir $X$ uzayında tanımlı $A$ ve $B$ bulanık kümelerinin, eğer her $x \in X$ için üyelik fonksiyonları eşitse, ilgili bulanık kümeler birbirlerine eşittir:

$$\forall x \in X: \mu_A(x) = \mu_B(x)$$

Ve eğer;

$$\forall x \in X: \mu_A(x) \leq \mu_B(x)$$

ise A kümesi $B$'nin alt kümesi ($A \subseteq B$) olarak ifade edilir.

İki bulanık küme $A$ ve $B$'nin kesişim ($\cap$) işlemi için, literatürde pek çok farklı tanımlama vardır. Genellikle seçilmesi gereken tanımlama, uygulamaya göre seçilir. Örneğin;

$$\forall x \in X: \mu_{A\cap B} = \mu_A(x) \wedge \mu_B(x) = \{\min(\mu_A(x), \mu_B(x)\}$$

$$\forall x \in X: \mu_{A\cap B} = \mu_A(x) \wedge \mu_B(x) = \begin{cases} \dfrac{\mu_A(x) + \mu_B(x)}{2} \end{cases}$$

$$\forall x \in X: \mu_{A\cap B} = \mu_A(x) \wedge \mu_B(x) = \{\mu_A(x)\mu_B(x)$$

şeklindedir. Ayrıca iki bulanık $A$ ve $B$ kümesinin birleşim ($\cup$) işlemi de farklı yollarla tanımlanabilir. Birleşim için verilebilecek bazı farklı örneklerde aşağıdaki gibidir.

$$\forall x \in X: \mu_{A\cup B} = \mu_A(x) \vee \mu_B(x) = \{\max(\mu_A(x), \mu_B(x))$$

$$x \in X: \mu_{A\cup B} = \mu_A(x) \vee \mu_B(x) = \begin{cases} \dfrac{2\min\big(\mu_A(x), \mu_B(x)\big) + 4\max(\mu_A(x), \mu_B(x))}{6} \end{cases}$$

$$x \in X: \mu_{A\cup B} = \mu_A(x) \vee \mu_B(x) = \{\mu_A(x) + \mu_B(x) - \mu_A(x)\mu_B(x)$$

Bir $A$ bulanık kümesinin tümleyeni $A'$ ise aşağıdaki gibi tanımlanır.

$$\forall x \in X: \mu_{A'}(x) = 1 - \mu_A(x)$$

### 4.1.4. Dönüşüm Operatörleri

Dönüşüm operatörleri, bulanık kümeyi bir dilsel terimin ifade ettiği biçimde değiştirmek için üyelik fonksiyonları üzerinde çalışırlar. Örneğin, "çok yaklaşık 10" ifadesinde, "yaklaşık 10" bulanık kümesi "çok" dilsel terimi ile dönüştürülür. Bu ve benzeri operatörlerden bazıları Çizelge 4.1.'de verilmiştir.

**Çizelge 4.1.** Dönüşüm operatörü örnekleri

| Dilsel Terim | Üyelik Fonksiyonu Üzerindeki Değişim |
|---|---|
| Çok | $\mu_{\tilde{A}}(x) = (\mu_A(x))^n \; n > 1$ |
| Fazla veya Az | $\mu_{\tilde{A}}(x) = (\mu_A(x))^n \;\; 0 < n < 1$ |

### 4.1.5. Bulanık Kümelerde Kartezyen İç Çarpımı

$A_1, A_2, \dots, A_v$ bulanık kümeleri $U_1, U_2, \dots, U_v$ uzaylarında tanımlı olduğu varsayıldığında, bu kümelerin Kartezyen iç çarpımı, $U_1 \chi U_2 \chi \dots \chi U_v$ uzayında tanımlı bir $F = A_1 \chi A_2 \chi \dots \chi A_v$ bulanık kümesidir. Ve bu kümeye ait üyelik fonksiyonu da $\mu_F(u_1, u_2, \dots, u_v) = \bigcap_{i=1}^{v} \mu_{A_i}(u_i)$ şeklinde olacaktır.

### 4.1.6. Bulanık İlişkiler

$U_1$ ve $U_2$ ilgili uzay ve üyelik fonksiyonu $\mu_R: U_1 \chi U_2 \rightarrow [0,1]$ olsun. Bu durumda bulanık ilişki R, $U_1$ ve $U_2$ üzerinde tanımlıdır. $U_1$ ve $U_2$ sürekli olduğunda;

$$R = \int \mu_R \frac{(u_1, u_2)}{(u_1, u_2)}$$

kesikli olduğunda ise;

$$R = \sum \mu_R \frac{(u_1, u_2)}{(u_1, u_2)}$$

şeklinde olur (Zimmermann, 1996).

### 4.1.7. Bulanık Küme Bileşimi

$R_1$ ve $R_2$ sırası ile $U_1 \chi U_2$ ve $U_2 \chi U_3$ üzerinde tanımlı iki bulanık ilişki olsun. Böylece, $R_1$ ve $R_2$ bileşimi olan $C$ bulanık ilişkisi aşağıdaki gibi tanımlanır (Papis and Siettos, 2005).

$$C = R_1 . R_2 = \left\{(u_1, u_3) \cup \left(\mu_{R_1}(u_1, u_2) \cap \mu_{R_2}(u_2, u_3)\right)\right\}$$

$$u_1 \in U_1, u_2 \in U_2, u_3 \in U_3$$

### 4.1.8. Bulanık Gerektirme

$A$ ve $B$ sırası ile $U_1$ ve $U_2$ üzerinde tanımlı iki bulanık küme olsun. Gerektirme $I: A \rightarrow B \in U_1 \chi U_2$ aşağıda ki gibi tanımlanmıştır (Ross, 1995; Zimmermann, 1996):

$$I = A \chi B = \int \frac{\mu_A(u_1) \cap \mu_B(u_2)}{(u_1, u_2)}$$

"Eğer hata negatif büyük ise kontrol çıktısı pozitif büyüktür." kuralı, aslında kontrol çıktısının y olması hatanın x olmasını gerektirir, gerektirmesidir.

Varsayalım ki, iki ayrık bulanık küme $A = \{(u_i, \mu_A(u_i), i = 1, \dots, n)$ ve $B = \{(v_j, \mu_B(v_j), j = 1, \dots, m)$, sırası ile $U$ ve $V$ üzerinde tanımlı olsun. Böylece $A \rightarrow B$ gerektirmesi $U$ ve $V$, üzerinde tanımlı R bulanık ilişkisi olacaktır.

$$R = \left\{\left((u_i, v_j)\left(\mu_R(u_i, v_j)\right)\right), i = 1, \dots, n, j = 1, \dots, m\right\}$$

Burada $\mu_R(u_i, v_j)$ ise;

$$\begin{bmatrix} \mu_A(u_1) \\ \mu_A(u_2) \\ \dots \\ \mu_A(u_n) \end{bmatrix} \chi [\mu_B(v_1) \quad \mu_B(v_2) \quad \dots \quad \mu_B(v_m)\ ]$$

$$= \begin{bmatrix} \mu_A(u_1) \wedge \mu_B(v_1) & \cdots & \mu_A(u_1) \wedge \mu_B(v_m) \\ \vdots & \ddots & \vdots \\ \mu_A(u_n) \wedge \mu_B(v_1) & \cdots & \mu_A(u_n) \wedge \mu_B(v_m) \end{bmatrix}$$

şeklinde olacaktır.

### 4.1.8. Çıkarsama Kuralları

Temelde çıkarsama var olan bilgiden yaralanarak, yeni bilgi elde etmeye denir. Varsayalım ki, $R$, $U_1 \chi U_2$ üzerinde tanımlı bir bulanık ilişki ve $A$, $U_1$ üzerinde tanımlı bir bulanık küme olsun. $A.R = B$ bileşimi, $U_2$ üzerinde tanımlı bir bulanık kümedir. Bu küme, $A$ (durum) bulanık kümesinden, $R$ (kural) gerektirmesi temel alınarak yapılarak, sonuç durumunu temsil etmektedir (Papis and Siettos, 2005).

$N$ kuraldan oluşan bir kural tabanı olduğunda ve $i$. kural şöyle verildiğinde,

$$Eğer\ A_{i2}\ ve\ A_{i2}\ ve\ \dots ve\ A_{in}\ ise\ B$$

burada, $x_i$ girdi değişkeni sayısı $n$, i. kurala ait $x_j$ girdi değişkenin bulanık kümesi $A_{ij}$ ve i. kurala ait $y_i$ çıktı değişkenin bulanık kümesi $B_i$ olacaktır. i. kurala ait gerektirme ise;

$$I_i = A_i \rightarrow B_i, A_i = A_{i1} \cap A_{i2} \cap \dots \cap A_{in} = \bigcap_{i=1}^{n} A_{ij}$$

olacaktır. Böyle tüm kuralara ait gerektirme;

$$I_{tüm} = R_1 \cup R_i \cup \dots \cup R_n = \bigcup_{i=1}^{n} R_i = \bigcup_{i=1}^{n} A_i \rightarrow B_i$$

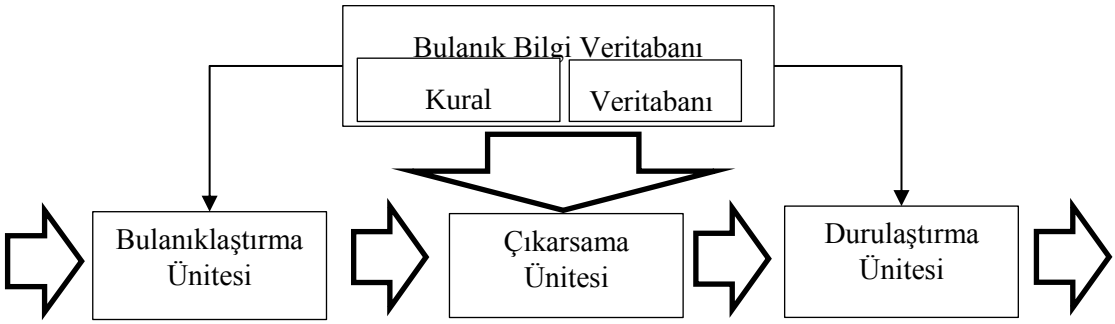
olacaktır.

## 4.2. Bulanık Çıkarsama Sistemi

Fiziksel sistemleri tanımlarken, nitel ve nicel olarak, matematiksel formülasyon kullanılır. Bu formülasyon, matematiksel model olarak adlandırılır. Fakat çoğu gerçek sistemin; özellikle karmaşık olanların, böyle bir modelle tanımlanması neredeyse imkânsızdır. Bunun nedenlerinden bazıları; sistemin

karmaşıklığı, kesin olmaması, lineer olmaması ve rasgelelik içermesi vb. olabilir. Bu nedenle, özellikle gerçek hayat uygulamalarında, yaklaşık modelleme hem gerekli hem de yararlı bir yaklaşımdır. Ayrıca yaklaşık modelleme her durumda mümkün olabilir. Fakat buradaki asıl soru nasıl bir yaklaşık modellemenin yeterince iyi olduğudur. Buradaki yeterince iyi sorusunun cevabı modellenen sistemin verdiği cevapların, modeli kullanan tarafında yeterli bulunması ile ilişkilidir. Aslında karmaşık sistemlere verilecek en belirgin örnek, insan bilgisinin temsilidir. Ve bu temsili sağlayacak en uygun yol ise bu bilgiyi dilsel terimlerle açıklamaktır. Böyle bir bilgi temsilinin modellemesi için başvurulan yollardan biri de bulanık mantıktır.

Bulanık çıkarsama eldeki verilerin bulanık mantık kullanılarak çıktıya dönüştürülme sürecidir. Literatürde farklı şekillerde de isimlendirilirler: Bulanık Kural Tabanlı Sistemler, Bulanık Uzman Sistemler, Bulanık Model, Bulanık İlişkilendirilebilir Bellek, Bulanık Mantık Denetleyici, Bulanık Sistem gibi.




**Şekil 4.2.** Bulanık çıkarsama sisteminin temel yapısı

Bulanık çıkarsama sisteminin temel yap"ısında; bulanıklaştırma ünitesi, çıkarsama ünitesi, bilgi tabanı ve durulaştırma ünitesi (Şekil 4.2.) bulunur. En önemli elemanı çıkarsama ünitesidir. Bu ünitenin sahip olduğu iki ana bilgi tipi vardır.

1. Veritabanı; sistem değişkeni olarak kullanılan her bir bulanık kümenin üyelik fonksiyonlarının tiplerini, sayılarını ve etiketlerini tanımlar. Bunlar girdi ve çıktı değişkenleri olarak kategorize edilir. Tasarımcı her biri için uygun bir bulanık küme tanımlar. Bu seçilimin doğru yapılması tasarımın en kritik adımıdır. Ve bu adımda yapılacak ufak bir hata dahi, sistemin performansında ciddi etkiler doğurabilir. Her bir değişkenin bulanık kümesinin tanım uzayı değişkenle aynıdır.

2. Kural tabanı, temelde girdi bulanık değerlerini, çıktı bulanık değerleri ile ilişkilendirir. Bu aslında karar verme tarzını yansıtır. Kontrol stratejisi kural tabanında saklanır. Bu aslında, bulanık kontrol kuralları kümesidir. Bulanık kurallar, kural tabanında ifade edilen kontrol ilişkilerini "Eğer-ise" formatında birleştirir. Örneğin; iki girdi, bir çıktıya sahip bir bulanık kural için genel hal aşağıdaki gibi olur.

$$Kural\ i: Eğer\ x, A_i\ ve\ y, B_i\ ise\ z, C_i' dir.$$

Yukardaki örnekte x ve y girdi değişkenleri, z ise çıktı değişkenleridir. $A_i$, $C_i$ ve $B_i$ kısımları ise "negatif", "pozitif", "sıfır" gibi dilsel terimlerdir (Bulanık kümeler). Eğer-ise kuralının "eğer" kısmına öncül önerme, "ise" kısmına ise sonuç önerme denir.

Genelde sisteme gelen veya sistemden istenen sonuçlar kesin değerlerdir. Bulanık çıkarsama sisteminin durulaştırma ve bulanıklaştırma adımı, sisteme gelen veya sistemden çıkan değerler için kesin değerler ve bulanık değerler arasında eşleştirme işlemini gerçekleştirir. Yani çıkarsama ünitesinin, kullanacağı değerlerin veya çıkarttığı sonuçların kullanıma hazır hale getirilmesi görevini üstlenir.

Çıkarsama ünitesi ise, verilen bulanık girdiden bir çıktı elde etmek için, çeşitli bulanık mantık işlemlerini gerçekleştirir. Bulanık çıkarsama boyunca, sırası ile aşağıdaki işlemler gerçekleşir:

1. Her bir sistem girdisi için, bulanık girdi verisi ile tanımlanmış bulanık kümeler arası eşleşme derecesi belirlenmesi.
2. Her bir kural için, öncül önermesindeki girdi değişkenleri ile eşleşme derecelerine ve bağlantılarına (ve, veya) dayalı ateşleme gücü (ilgi ve uygulanabilirlik derecesi) hesaplanması.
3. Her bir kuralın sonuç önermesindeki çıktı değişkenleri için tanımlanmış bulanık kümeler ve hesaplanmış ateşleme gücü ile çıktıların türetilmesi.

Kural tabanına dayalı bulanık çıktı çıkarımı ile ilgili önerilmiş pek çok teknik vardır. Bunlarda en çok kullanılanları, "Max – Min" bulanık çıkarsama metodu ve "Max – Product" bulanık çıkarsama metodudur ( Mamdani, 1976; Larsen, 1980).

"Max – Min" çıkarsama metodunda; "Bulanık Ve" operatörü (kesişim), öncüllerdeki minimum değerin alınacağını;

$$\mu_A\ VE\ \mu_B = \min\{\mu_A, \mu_B\}$$

söylerken, “Max – Product” metodu, öncüllerin çarpımının alınacağını söyler.

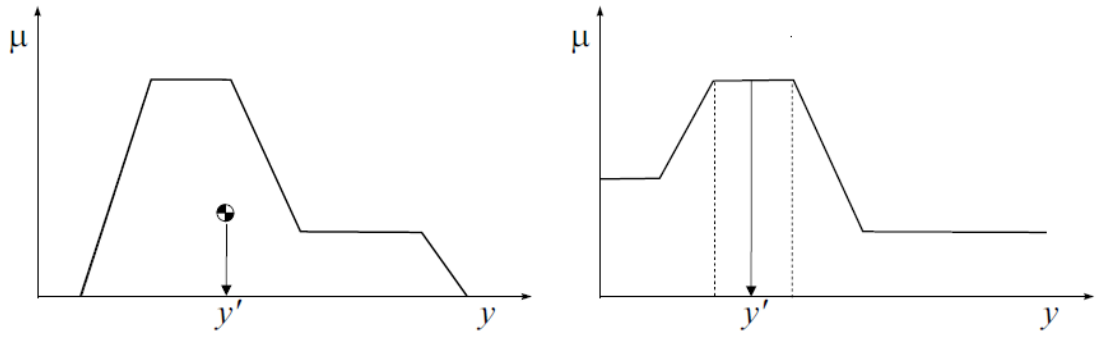
$$\mu_A \, VE \, \mu_B = \mu_A \mu_B$$

Burada, herhangi iki $\mu_A$, $\mu_B$ üyelik fonksiyonu sırası ile $A$ ve $B$ bulanık altkümelerine ait olsun. Tüm kurallardan gelen bilgiler, birleşim operatörü yardımıyla, birleştirilir ve böylece $C$ bulanık çıktı uzayı elde edilir.

### 4.2.1. Durulaştırma Ünitesi

Çıkarsama ünitesinden çıkan bulanık sonucu, tanımlanmış üyelik fonksiyonlarını kullanarak kesin sonuca dönüştürme aşamasıdır.

Literatürde pek çok farklı durulaştırma metodu vardır. Her biri, ilgilenilen sistemin gereksinimlerini karşılamak için farklı stratejiler izler. Bir metodun diğerinden kesin üstünlüğü gibi bir durum söz konusu değildir. Çünkü ilgilenilen sistemin özeliklerinden, sistem tasarımcısının tercihlerine kadar pek çok farklı değerlendirme kriteri vardır. En çok kullanılan durulaştırma metotları; “Maksimumun Ortalaması” ve “Centroid” olarak ele alınabilir (Ross, 1995;Lee, 1990).




**Şekil 4.3.** (a) Centroid, (b) maksimumun ortalaması

Centroid tekniğinde kullanılan son bulanık uzayın ağırlık merkezi alınarak tek bir kesin sonuç elde edilir. Bu sonucu elde etmek için kullanılan formül;

$$y' = cog = \frac{\sum_{j=1}^{F} \mu_{B'}(y_j)\, y_j}{\sum_{j=1}^{F} \mu_{B'}(y_j)}$$

şeklinde verilir. Maksimumun ortalaması tekniğinde ise aşağıda verilen formül kullanılır.

$$mom(B') = cog\{y | \mu_{B'}(y) = \max_{y \in Y} \mu_{B'}(y)\}$$

### 4.2.2. Kural Tabanının Tasarımı

Kural tabanlarının tasarımlarında iki temel yaklaşım vardır (Yan et al., 1994). Bunlar sezgisel yaklaşım ve sistematik yaklaşımdır.

Sezgisel yaklaşım, arzu edilen çıktı cevabını almak amacıyla tasarlanan bulanık kontrol kuralarının yapımında, yalnız çalışılan sistemin davranışları hakkında nitel bilgi gerektiren, kullanışlı bir yol sağlar. İki girdi (*a* ve *b*) ve bir çıktı (*c*) 'dan oluşan sistemin kural tabanının tasarımı için bu sistemi tanıyan veya sistem üzerinde bir miktar gözlem yapıp deneyim sahibi olmuş bir sistem tasarımcısı olması yeterlidir. Tasarımcı tarafından, sistem ile ilgili nitel bilgiye dayalı ve dilsel terimlerle tasarlanmış bir kural tabanı oluşturulur. Çizelge 4.2. 'de görüleceği üzere eksenlerde bulunan dilsel terimler girdi değişkenleri iken matrisin her bir elemanı çıktı değişkenlerini yansıtır.

Sistematik yaklaşımda ise kural tabanı, sistem tanımlama, veri madenciliği ve örüntü tanıma teknikleri gibi teknikler yardımıyla oluşturulur.

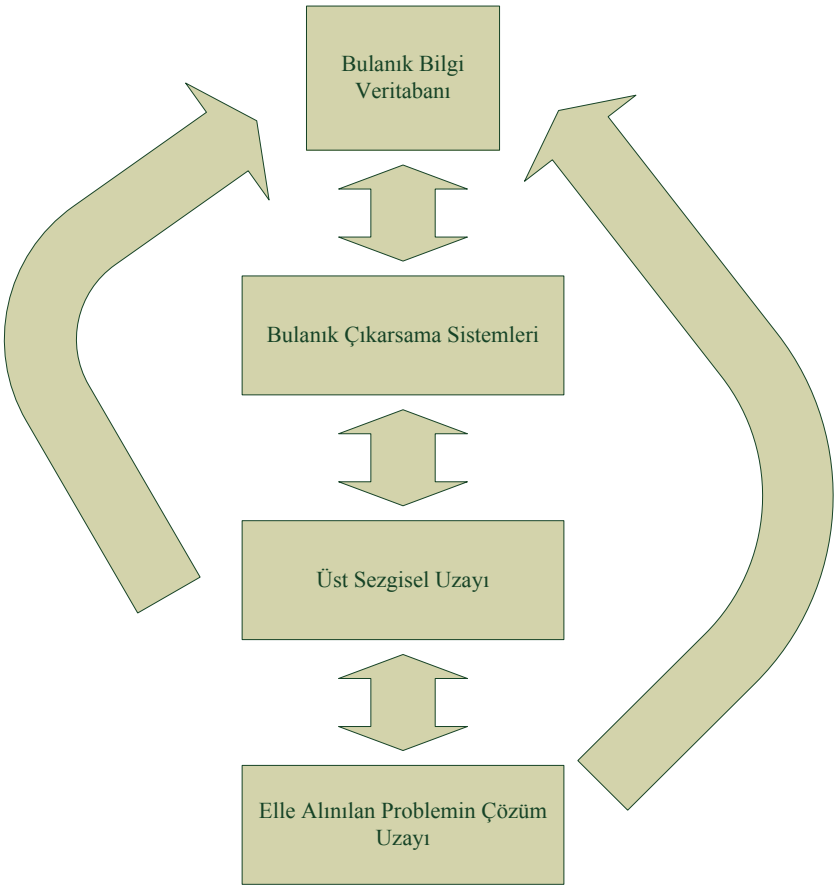
**Çizelge 4.2.** İki girdi bir çıktılı bir bulanık kural tabanı

| | | b | | | | |
|---|---|---|---|---|---|---|
| | | Negatif Büyük(NB) | Negatif Küçük(NK) | Sıfır(S) | Pozitif Küçük(PK) | Pozitif Büyük |
| a | Negatif Büyük(NB | PB | PB | PB | PK | S |
| | Negatif Küçük(NK) | PB | PB | PK | S | NK |
| | Sıfır(S) | PB | PK | S | NK | NB |
| | Pozitif Küçük(PK) | PK | S | NK | NB | NB |
| | Pozitif Büyük | S | NK | NB | NB | NB |

# 5. İŞBİRLİKÇİ ÜST SEZGİSEL SİSTEMİ

## 5.1. Sistemin genel şeması

Genel hatları ile sistem, tek bir bulanık bilgi veri tabanına sahip, birden çok bulanık çıkarsama sisteminin kullanıldığı ve üç ana probleme cevap veren bir sistemdir (Şekil 5.1).




**Şekil 5.1.** Sistemin genel şeması

Sistemin tasarım aşamasında önce bulanık bilgi veri tabanı, üst sezgisel uzayını oluşturan üst sezgiseller ve sistemin çözmesi arzu edilen problemin çözüm uzayı göz önünde bulundurularak ve içeriği daha sonra detaylandırılacak bulanık çıkarsama sistemlerinin ihtiyaçlarına karşılık verecek şekilde oluşturulmalıdır.

Bölüm 4.2.2.'de bahsedildiği üzere bunu gerçekleştirmenin iki yolu sezgisel veya sistemik olaraktır. Bu aşama, sistem için ön hazırlık aşaması gibi görülse de, hem oluşturulacak çıkarsama sistemleri ile hem de genel çözümün performansı ile doğrudan ilişkili olduğu için kritik öneme sahiptir.

İlk olarak ele alınan problem, ilgilenilen optimizasyon problemini hangi üst sezgisel ile (en azından başlangıç için) çözüleceği sorunudur. Bu nokta da problem özelinde değişiklik gösterecek şekilde, ele alınan problemin çözüm uzayına ait bir takım parametreler hesaplanarak başlanır. Daha sonrasında, bulunan parametreler, bulanık bilgi veri tabanına aktarılır. Elde edilen veriler, bulanık çıkarsama sistemi içindeki ilgili bulanık çıkarsama sistemine gönderilerek, sistemin elindeki üst sezgisel uzayındaki, üst sezgisellerden birini seçmesi beklenir. Bu seçilen üst sezgisel ilgilenilen problemi çözmekte veya en azından belirli bir süre için çözmekte kullanılacak başlangıç üst sezgiseli olacaktır.

Buradan sonraki, ikinci aşama seçilmiş olunan üst sezgiselin kontrol parametrelerini belirleme aşaması olacaktır. Bu noktada yine; ele alınılan problemin çözüm uzayına ait, hesaplanmış ve bulanık bilgi tabanına aktarılmış belirli parametreler kullanılacaktır. Bulanık çıkarsama sistemlerinde, üst sezgisel uzayında bulunan her bir üst sezgiselin kontrol parametrelerinin belirlenmesi için görevlendirilmiş bulanık çıkarsama sistemleri olmalıdır. Böylece ilgili bulanık çıkarsama sisteminden kontrol parametrelerine karar vermesi istenir. Sonuçta, sistem problemin çözüm uzayında arama yapmaya başlayacak hale gelmiş olacaktır.

Çözüm uzayında arama aşamasına geçildikten sonra, arama işlemini gerçekleştiren üst sezgisel her iterasyonda veya önceden belirlenmiş belli zaman aralıklarında, seçili üst sezgisele göre değişecek şekilde, belirli parametreleri bulanık bilgi veri tabanına yollar. Bulanık bilgi veri tabanında, toplanan bu parametreler, en az iki ayrı bulanık çıkarsama sistemine yollanır. Burada iki temel soruya, cevap aranır: Üst sezgiselin başlangıç parametrelerine ayarlama yapmaya ihtiyaç var mı? Üst sezgisel, arama uzayında verim vermeye devam edebilir mi? Sistem bu iki soruyu sırası ile cevaplandırmaya çalışacaktır.

İlk soru ele alındığında, sisteme dâhil olan her bir üst sezgisel için farklılık gösterecek bazı parametre hesapları yapılır. Bu hesaplanmış olan parametreler, bulanık çıkarsama sistemlerinde ilgili sistemlere aktarılır. Burada, üst sezgiselin

başlangıç parametrelerinin yeterli olduğu veya belirli değişikliklere tabi tutulması gerekli olduğuna dair bir karar verilir. Ve buna göre sistem, çalışan üst sezgisele geri bildirimde bulunarak parametrelerinde ayarlama yapılmasını sağlar. Böylece üst sezgiselin çözüm uzayında yapacağı aramanın geri kalanının daha iyi performans göstermesi için gerekli parametre ayarlamasının ne zaman ve nasıl yapılacağı soruları cevap bulmuş olur.

İkinci soru ise, üst sezgisel çözüm uzayının keşfedilmesi konusunda umut vaat edici davranıp davranmadığıdır. Bu noktada, çözüm uzayının keşfedilen kısmı ve üst sezgiselin o anki ve geçmiş davranışları ile ilgili bazı hesaplamalar yapılır. Bu hesaplamalar hem seçili üst sezgisel hem de ilgilenilen problem özelinde farklılık gösterebilir. Bu noktada, sürekli olarak hesaplanarak bulanık bilgi veri tabanına aktarılan bu bilgiler, buradan bu konuda karar vermesi beklenen ilgili bulanık çıkarsama sistemine gönderilir. Bu sistem, üst sezgiselin eldeki çözüm uzayı için devam ya da tamam kararını alacaktır.

Ancak bu aşamada yeni bir sorun ortaya çıkmış olur şayet sistem, üst sezgiselin artık yeterli olmadığına karar verirse ne yapılmalıdır. Bu soruya da yine sistem kendi içinde cevap verecektir. Bulanık çıkarsama sistemlerinin içinde; bir üst sezgisel bırakılacağı zaman, çözüm uzayını hangi üst sezgiselin araması gerektiğine dair karar verme görevini üstlenmiş bulanık çıkarsama sistemleri vardır. Bu sistemlerden ilgili olanı, bırakılan üst sezgisel yerine kimin geçeceğine karar verecektir. Bu karar sonucunda seçilen üst sezgiselin de parametreleri belirlenmesi ihtiyacı doğmuş olacaktır. Bu ihtiyaca karşılık vermek için yine, belirli bulanık çıkarsama sistemleri devreye girerek üst sezgiselin başlangıç parametrelilerini ayarlayacaktır. Bu aradaki önemli nokta üst sezgisel uzayındaki bir diğer üst sezgisele geçerken, yeni gelen üst sezgiselin çözüm uzayındaki aramaya kalınan yerden devam etmesidir. Böylece çözüm uzayının daha önceki üst sezgiselin kabiliyetleri doğrusunda hiç mercek altına alınmamış ya da kısmi bir incelemeye tabi tutulmuş olan kısımlarını da arama imkânı doğacaktır. Bu durumun farklı üst sezgisellerin arama sırasındaki eksikliklerinin, kapatılması konusunda yardımcı olacağı düşünülmektedir.

Bu noktadan sonra süreç, ortaya konan genel şema doğrultusunda devam edecektir. Ancak bu aşamada, son bir sorun ile karşılaşılmış olunur. Bu sorun, aramanın ne zaman sonlandırılacağı sorunundur. Arama boyunca, seçilmiş üst sezgisellerden bağımsız olarak çözüm uzayının keşfedilme durumu ile ilgili bazı bilgiler bulanık bilgi veri tabanına aktarılır. Bu bilgiler ile her iterasyonda veya

belirli zaman aralıklarında, aramanın sonlanıp, sonlanmayacağından görevli olan bulanık çıkarsama sistemi çalıştırılır. Böylece bu sistem, aramanın daha fazla umut vaat etmediği veya bunda soran yapılacak iyileştirilmelerini, arama maliyeti karşındaki marjinal faydasının yetersiz olduğu kanısına varırsa aramayı sonlandıracaktır.

## 5.2. 0-1 Sırt Çantası Problemi İçin Bir İşbirlikçi Üst Sezgisel Sistemin Tasarlanması

Bu bölümde, daha önce aşamaları genel hatları ile tarif edilen sistemin, 0-1 Sır Çantası Problemi üzerinde çalışan bir prototipi tasarlanacaktır. Her bir aşaması detaylandırılarak anlatılacaktır. Ancak sistemin tasarımını geçmeden önce, deneysel kurulum gerçekleştirilmelidir.

### 5.2.1. Deneysel Kurulum

Deneysel kurulumda, 30 adet problem rasgele olarak üretilmiştir. Bu problemler toplam eşya sayısı ($N$), eşya ağırlığı ve faydası için açıklık değeri ($R$) ve fayda ile ağırlık arası korelasyon tipi olmak üzere 3 parametrenin tüm mümkün kombinasyonları kullanılarak üretilmiştir. Burada $N$, 100 200 500 1000 ve 2000 olmak üzere 5 farklı büyüklükte kullanılmıştır. Açıklık değeri $R$ ise 1000 ve 10000 olmak üzere 2 farklı değer ile kullanılmıştır. Problemler arası en belirleyici farkı yaratan ise korelasyon tipidir. Çünkü genelde pek çok Sırt Çantası Probleminin seviyesinin büyük ölçüde korelasyona dayandığı görülür. 3 farklı korelasyon tipinin yeterli olduğu literatürden anlaşılmıştır (Martello and Toth, 1990). Burada;

- Korelasyonsuz: $w_j$ ve $p_j$, $[1, R]$ arasında tekdüze rasgele olarak dağılır.
- Zayıf korelasyonlu: $w_j$ , $[1, R]$ arasında tekdüze rasgele olarak dağılırken; $p_j$, $[w_j - R/2, w_j + R/2]$ arasında tekdüze rasgele olarak dağılır.
- Korelasyonlu: $w_j$ , $[1, R]$ arasında tekdüze rasgele olarak dağılırken; $p_j = w_j + (\frac{R}{10})$.

Oluşturulan her bir problem için kapasite $c = uniform(0,1) * \sum_{j=1}^{n} w_j$ şeklinde hesaplanmıştır.

Korelasyonun artması demek, $max_j(^{p_j}/_{w_j})$ ile $min_j(^{p_j}/_{w_j})$ arasında ki farkın azalması demektir. Bu durumda problemin beklenen zorluk seviyesinde artış

gözlenmektedir. Literatür deneyimlerinden yola çıkılırsa, zayıf korelasyonlu problemlerin gerçek hayat durumlarına en yakın olduğu sonucuna erişilir (Martello and Toth, 1990).

### 5.2.2 Sistemin Kurulumu

Deneysel kurulum aşamasında oluşturulmuş olan 30 adet problem, farklı parametre kombinasyonları ile Tavlama Benzetimi, Tabu Araması, Değişken Komşuluk Araması, Genetik Algoritma, Karınca Kolonisi Optimizasyonu ve Yapay Arı Kolonisi algoritması ile çözülmüştür. Sonuçlar ele alındığında, pek çok durumda Tavlama Benzetimi, Genetik Algoritma ve Yapay Arının daha iyi çözümler verdiği görülmüş ve bu görüş yapılan varyans analizi ile desteklenmiştir. Elde edilen sonuçlara sezgisel olarak bakılarak, Sırt Çantası Probleminin çözümü için kullanılması gereken üst sezgisellerin Tavlama Benzetimi, Genetik Algoritma ve Yapay Arı Kolonisi algoritması olduğuna karar verilmiştir. Belirlen bu üst sezgisellerin standart halleri uygulanmış ve herhangi bir farklılaştırma gerçekleştirilmemiştir. Seçim esnasında üst sezgiselleri yörünge tabanlı ve popülasyon tabanlı olarak sınıflandırılmasına dikkat edilmiştir. Tüm üst sezgiseller sadece amaç fonksiyonu değeri ile yönlendirilmiştir.

Üst sezgisellerin seçilimi gerçekleştirdikten sonra, daha önce çözümü gerçekleştirilmiş problemlerle oluşturulmuş veri tabanı, literatür bilgileri ve uzman görüşü ışığında bulanık bilgi tabanının oluşturulmasına geçilmiştir. Burada kurallar, girdi ve çıktı değişkenleri ve bu değişkenlere ait üyelik fonksiyonları da bulunacaktır. Kurallar belirlenmeden önce bulanık çıkarsama sistemlerinin genel yapısı ortaya konmalıdır. Tasarlanan sistemdeki tüm bulanık çıkarsama sistemleri Çizelge 5.1 de gösterildiği gibidir.

**Çizelge 5.1.** Bulanık Çıkarsama Sistemleri Tasarımı

| | |
|---|---|
| Ve Metodu | Minimum |
| Veya Metodu | Maksimum |
| Çıkarsama | Minimum |
| Toplama | Maksimum |
| Durulaştırma | Maksimumun ortalaması |

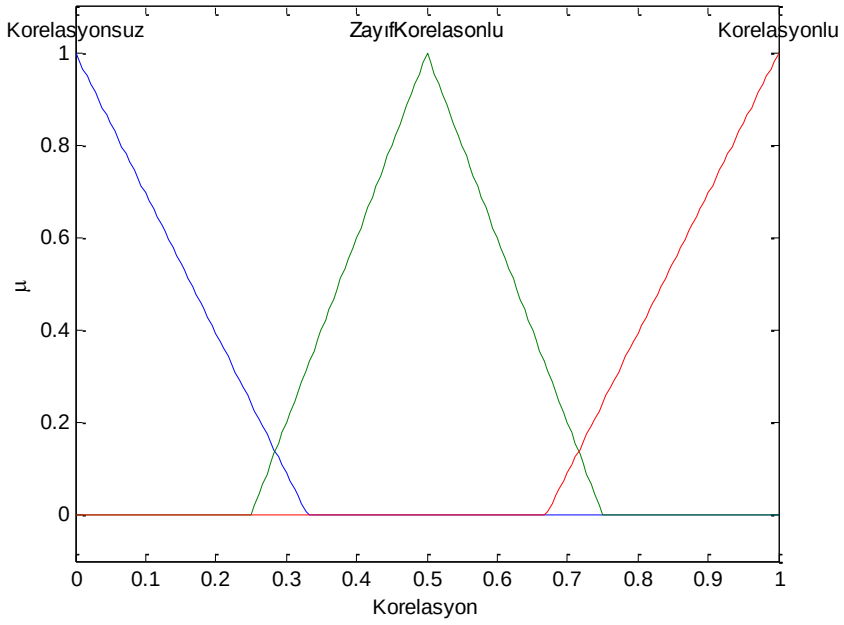
Daha önce verilen şema doğrultusunda ilk aşamada oluşturulması gereken kurallar, ilgilenilen optimizasyon problemini hangi üst sezgisel ile çözülmeye

başlanacağına karar verilmesi gerektiği ile ilgili olanlardır. Burada sistemin başlangıç anında ele alınan problemin, çözüme girmeye aday bileşenlerinin bulunduğu uzay ile ilgili, boyut ve korelasyon parametreleri hesaplanır. Bu hesaplanan parametreler ile Çizelge 5.2'de verilen kurallara göre çözüme hangi algoritma ile başlanılacağına karar verilir.

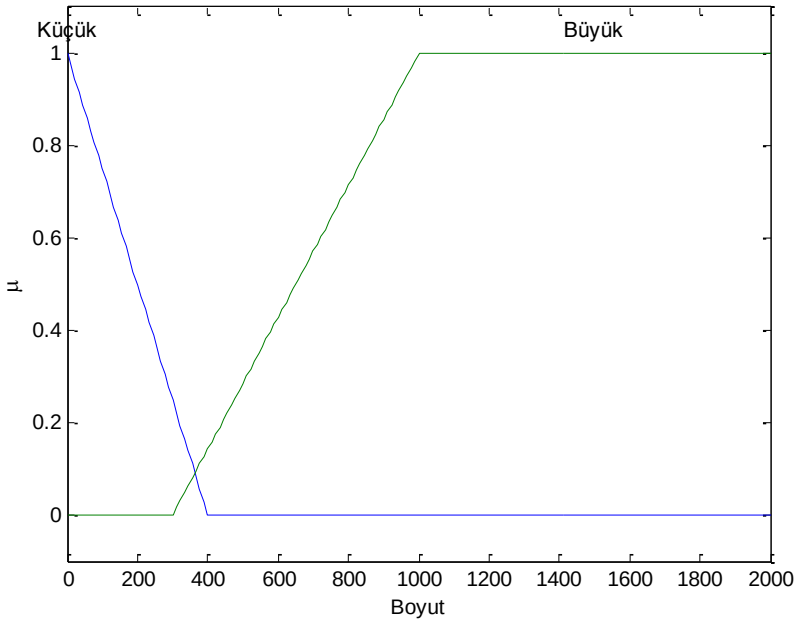
**Çizelge 5.2.** Başlangıç algoritması seçim kural tablosu

| Başlangıç Algoritması | | |
|---|---|---|
| Problem Boyutu / Korelasyon | Küçük | Büyük |
| Korelasyonsuz | SA | SA |
| Zayıf Korelasyonlu | SA | SA |
| Korelasyonlu | GA | SA |

Oluşturulan kuralardaki girdi değişkenlerine ait üyelik fonksiyonları 'da Şekil 5.2 ve Şekil 5.3' deki gibi tasarlanmıştır.




**Şekil 5.2.** Korelasyon değişkenine ait üyelik fonksiyonları




**Şekil 5.3.** Boyut değişkenine ait üyelik fonksiyonları

Geçilecek olan ikinci aşama da seçilmiş olunan üst sezgiselin kontrol parametrelerinin belirlenmesi gerecektir. Üst sezgisellerin çalışma prensiplerinde ki farklılıklardan ötürü, her bir üst sezgisel için ayrı kurallar devreye girmelidir. Ancak sürecin ilerleyen aşamalarında algoritmalar arasında geçiş yapmayı kolaylaştırmak için tüm üst sezgisellerde aynı temsil tipi yani sabit uzunluklu ikili dizi kullanılmıştır. Kuralları ateşlemek için, boyut ve korelasyon parametreleri kullanılacaktır. Tüm üst sezgiseller için kuralar Çizelge 5.3'de ki gibidir. Girdi değişkenlerine ait üyelik fonksiyonları Şekil 5.2 ve Şekil 5.3' deki gibidir.

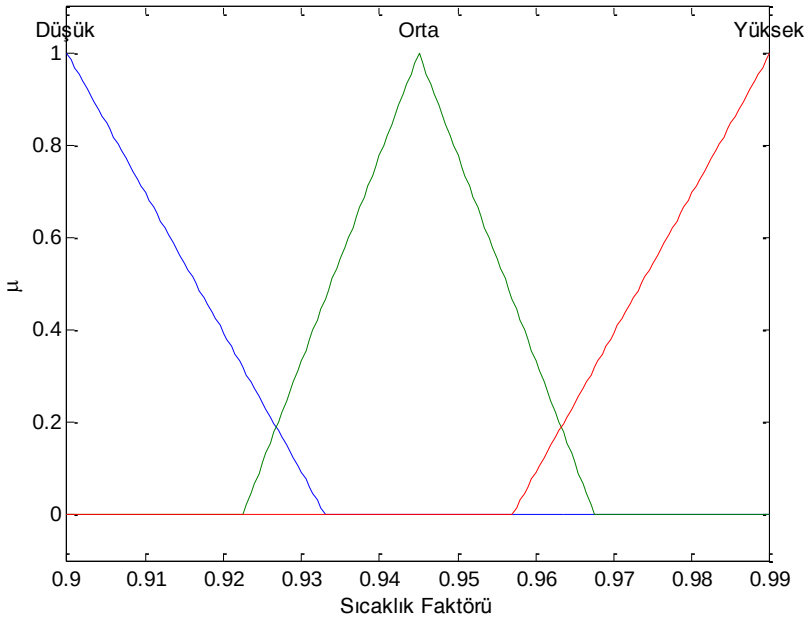
SA'nin kontrol parametreleri için kullanılacak kuralların çıktı değişkenleri başlangıç sıcaklığı (T) ve sıcaklık faktörü (SF)'dır. Bunlara ait üyelik fonksiyonları Şekil 5.4 ve Şekil 5.5'de ifade edildiği gibidir.

GA'nin kontrol parametreleri için kullanılacak kuralların çıktı değişkenleri popülasyon (N) ve Mutasyon olasılığı (Mo)'dır. GA için diğer tüm parametreler tüm süreç boyunca sabit kabul edilmiştir. Bunlara ait üyelik fonksiyonları Şekil 5.6 ve Şekil 5.7'de verilmiştir. Çaprazlama operatörü olarak 2 noktalı çaprazlama, çaprazlama olasılığı olarak 1.0 (Seçilen tüm çiftler çaprazlanmıştır.) ve seçim şeması olarak rulet tekerliği kullanılmıştır. Bu parametreler için seçilen değerlerin diğer tüm seçenekler üzerinde baskınlık gösterdiğine sezgisel olarak kanaat getirilmiş ve böylece sabitlenmiştir. Bunlara ek olarak %10'luk elitizim oranı kullanılmıştır.

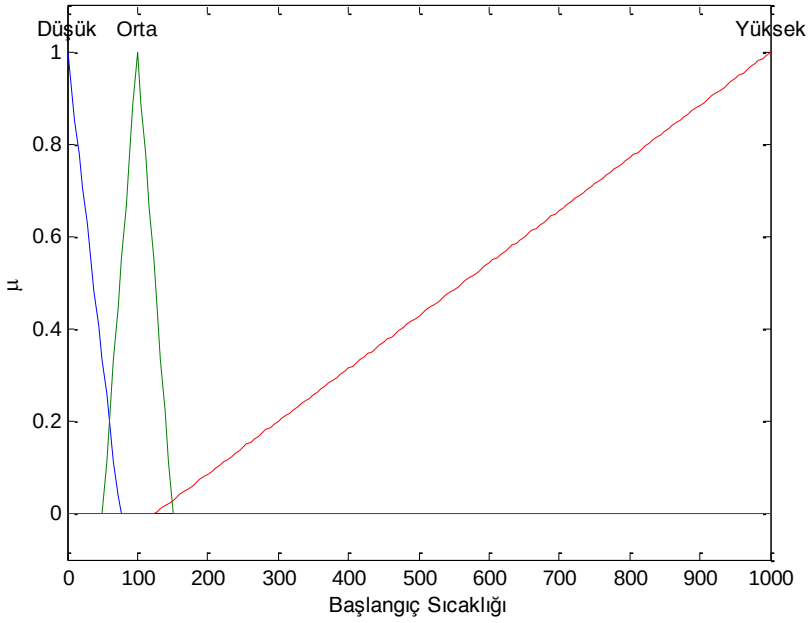
ABC'nin kontrol parametreleri için kullanılacak kuralların çıktı değişkenleri koloni boyutu (CS) ve limit (L)'dir. Koloni boyutu, görevli ve gözcü arıların toplam sayısını belirtmektedir. Görevli ve gözcü arı sayıları eşit olacak şekilde koloni boyutunun yarısı olarak belirlenmiştir. Bunlara ait üyelik fonksiyonları Şekil 5.8 ve Şekil 5.9'de gösterilmiştir.

**Çizelge 5.3.** Üst sezgisel kontrol parametresi belirleme kuralları

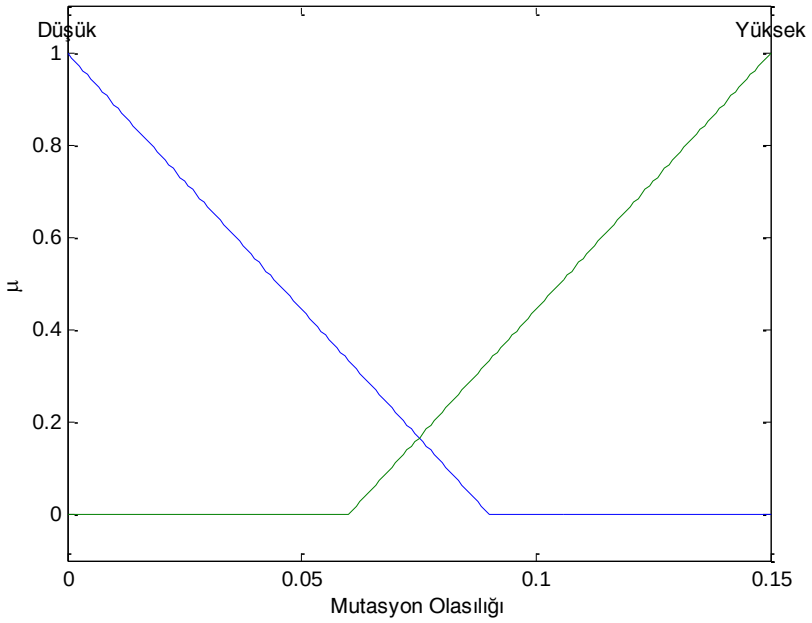
| SA Kontrol Parametreleri | | |
|---|---|---|
| | Boyut | |
| Korelasyon | Küçük | Büyük |
| Korelasyonsuz | T Orta SF Yüksek | T Yüksek SF Yüksek |
| Zayıf Korelasyonlu | T Orta SF Yüksek | T Yüksek SF Yüksek |
| Korelasyonlu | T Orta SF Yüksek | T Yüksek SF Yüksek |
| GA Kontrol Parametreleri | | |
| | Boyut | |
| Korelasyon | Küçük | Büyük |
| Korelasyonsuz | N Orta Mo Düşük | N Yüksek Mo Yüksek |
| Zayıf Korelasyonlu | N Orta Mo Düşük | N Yüksek Mo Yüksek |
| Korelasyonlu | N Orta. Mo Düşük | N Yüksek Mo Yüksek |
| ABC Kontrol Parametreleri | | |
| | Boyut | |
| Korelasyon | Küçük | Büyük |
| Korelasyonsuz | CS Düşük L Düşük | CS Yüksek L Yüksek |
| Zayıf Korelasyonlu | CS Orta L Orta | CS Yüksek L Yüksek |
| Korelasyonlu | CS Orta L Orta | CS Yüksek L Yüksek |



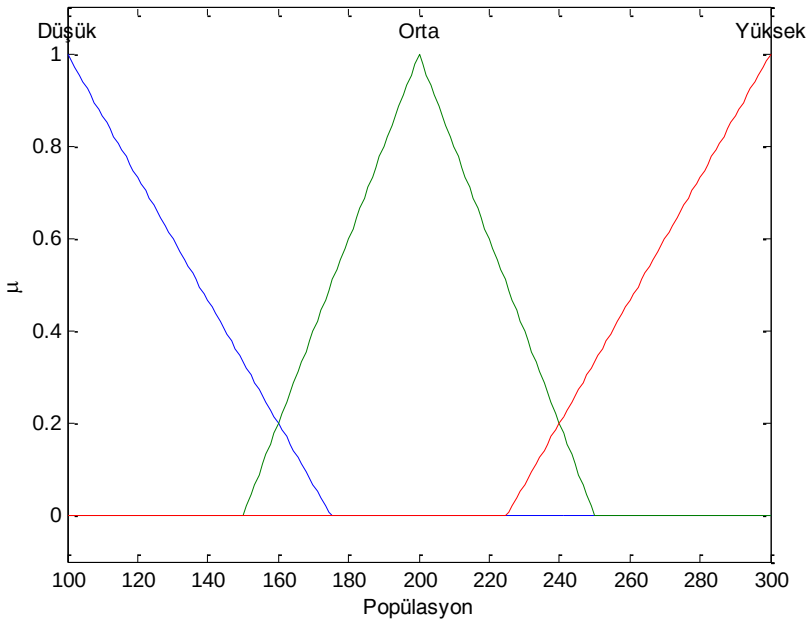

**Şekil 5.4**. Sıcaklık faktörü değişkenine ait üyelik fonksiyonları



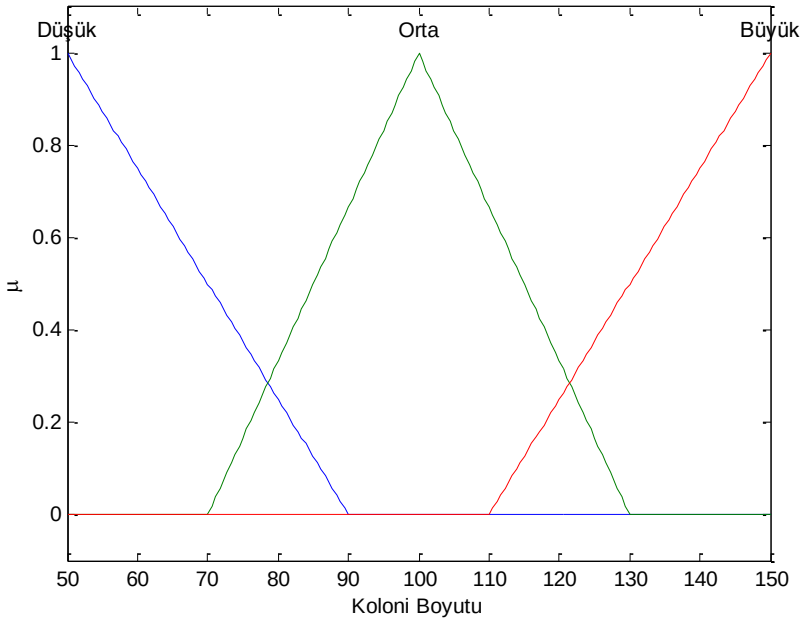

**Şekil 5.5.** Başlangıç sıcaklığı değişkenine ait üyelik fonksiyonları



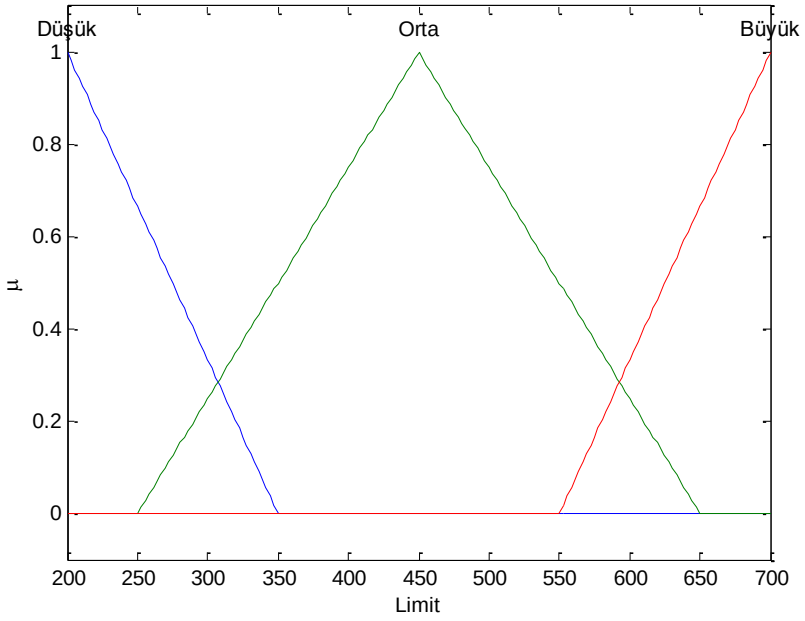

**Şekil 5.6.** Mutasyon olasılığı değişkenine ait üyelik fonksiyonları




**Şekil 5.7.** Popülasyon değişkenine ait üyelik fonksiyonları




**Şekil 5.8.** Koloni boyutu değişkenine ait üyelik fonksiyonları




**Şekil 5.9.** Limit değişkenine ait üyelik fonksiyonları

Sistemin genel şemasında daha önce bahsedildiği üzere, süreç başladıktan sonra, üst sezgiselin çalıştığı kontrol parametrelerinin yeterliliğine ve yeterli değilse nasıl değiştirilmesi gerektiğine karar verilmesi gerekmektedir. Bunun için üst sezgiselden elde edilecek veriler ile bir takım yeni parametre hesaplamaları gerekmektedir. Sistemin bu aşamasının oluşturulması sırasında daha önce tanımlanmamış bazı yeni değişkenler devreye girmektedir. Bunlardan ilki

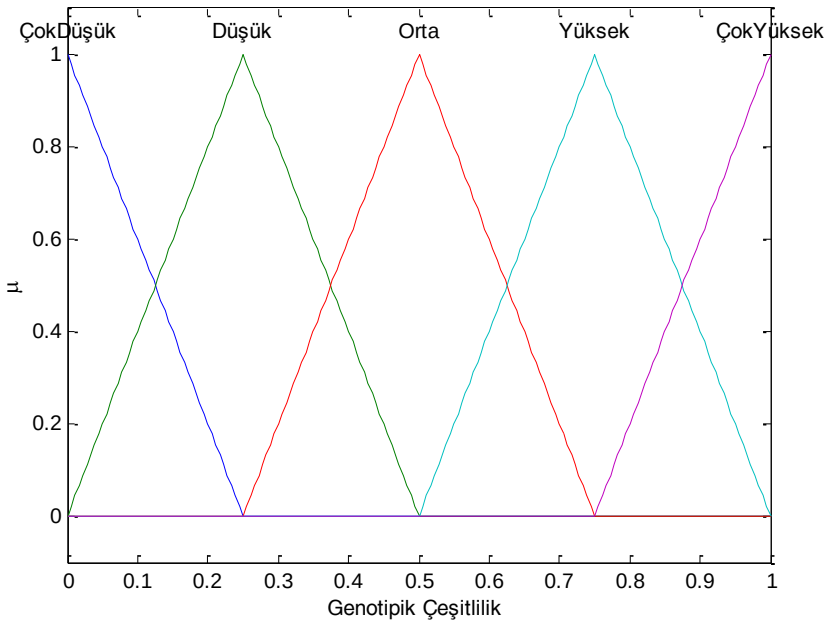
Normalize Edilmiş Ortalama Hamming Mesafesidir (NEOHM) ve tanımı aşağıdaki gibidir.

$$NEOHM = \frac{2}{n(n-1)} \sum_{i=1}^{n} \sum_{j=i+1}^{n} \frac{d_{ij}}{Genom\ uzunluğu}$$

Burada $d_{ij}$ , $i.$ ve $j.$ genom (sabit uzunluklu ikili dizi) arasındaki Hamming mesafesini (aynı uzunluktaki iki dizinin farklı elemanlarının sayısı) ifade etmektedir. Hesaplanan $NEOHM$ 'yi Genotipik Çeşitliliği (GÇ) temsil etmek için kullanılmıştır. Burada GÇ;

$$GÇ \in [0,1] == [Ç.Düşük, Ç.Yüksek]$$

şeklinde tanımlanmıştır. GÇ'ye ait üyelik fonksiyonları ise Şekil 5.10'daki gibidir.



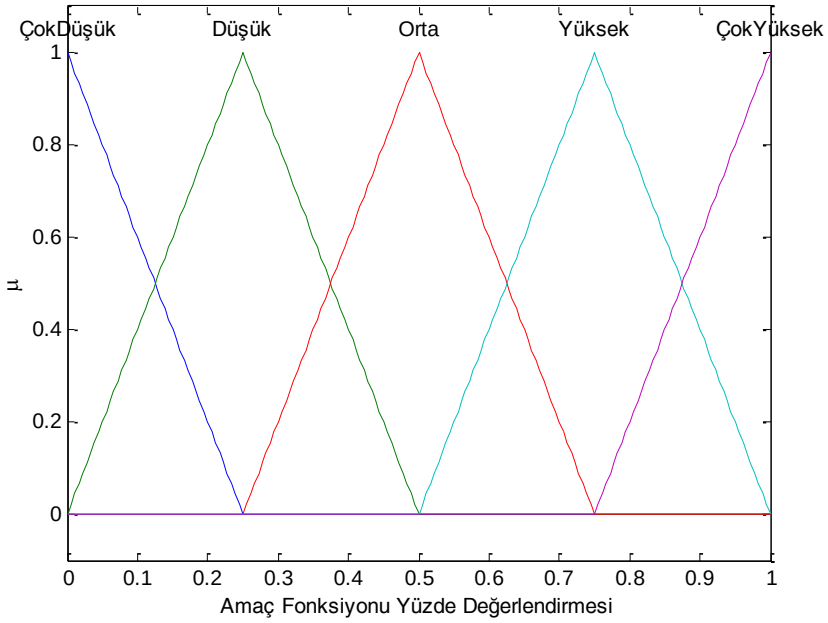

**Şekil 5.10.** Genotipik çeşitlilik değişkenine ait üyelik fonksiyonları

Kontrol parametrelerinin yeterliliği için hesaplanacak bir diğer parametre ise Amaç Fonksiyonu Yüzde Değerlendirmesi (AFYD) olacaktır. AFYD aşağıdaki gibi tanımlanmıştır.

$$AFYD = (Tamamlanan\ iterasyon\ sayısı)/(Toplam\ itersayon\ sayısı)$$

$$AFYD \in [0,1] == [Ç.Düşük, Ç.Yüksek]$$

Tamamlanan iterasyon sayısı, üst sezgiselin o ana kadar gerçekleştirdiği iterasyon sayıdır. Toplam iterasyon ise üst sezgisel özelinde belirlenmiş bir sabittir. Bu aşamada bu parametre yalnızca GA için kullanılacaktır. GA için toplam iterasyon sabiti 200 olarak ayarlanmıştır. Ayrıca GA'nin bu iterasyon sayısını geçmesi yasaklanmıştır. Yasaklanma durumuna ilerde tekrar değinilecektir. AFYD'ye ait üyelik fonksiyonu ise Şekil 5.11' da verilmiştir.




**Şekil 5.11.** Amaç fonksiyonu yüzde değerlendirmesi değişkenine ait üyelik fonksiyonları

Yukarıda bahsedilen parametreler ışığında, bu aşama için gerekli kurallar Çizelge 5.4 'de verildiği gibidir. Üst sezgisel kontrol parametrelerinin yeterliliği kuralları yalnız GA ve ABC için çalıştırılmaktadır. SA'nin kontrol parametrelerine algoritma çalışırken müdahale edilmemektedir.

**Çizelge 5.4.** Üst sezgisel kontrol parametrelerinin değişim oranı kuralları

| GA Parametreleri | | | | | |
|---|---|---|---|---|---|
| GÇ, AFYD→ ΔN | Amaç Fonksiyonu Yüzde Değerlendirmesi (AFYD) | | | | |
| Genotipik Çeşitlilik(GD) | Çok Düşük | Düşük | Orta | Yüksek | Çok Yüksek |
| Çok Düşük | Poz. Yüksek | Poz. Yüksek | Poz. Yüksek | Poz. Orta | Değişim Yok |
| Düşük | Poz. Yüksek | Poz. Yüksek | Poz. Orta | Değişim Yok | Değişim Yok |
| Orta | Poz. Yüksek | Poz. Orta | Değişim Yok | Değişim Yok | Neg. Orta |
| Yüksek | Poz. Orta | Değişim Yok | Değişim Yok | Neg. Orta | Neg. Orta |
| Çok Yüksek | Değişim Yok | Değişim Yok | Neg. Orta | Neg. Orta | Neg Yüksek |
| GÇ, AFYD→ ΔMo | Amaç Fonksiyonu Yüzde Değerlendirmesi (AFYD) | | | | |
| Genotipik Çeşitlilik(GD) | Çok Düşük | Düşük | Orta | Yüksek | Çok Yüksek |
| Çok Düşük | Poz. Yüksek | Poz. Yüksek | Poz. Orta | Poz. Orta | Değişim Yok |
| Düşük | Poz. Yüksek | Poz. Orta | Poz. Orta | Değişim Yok | Değişim Yok |
| Orta | Poz. Orta | Poz. Orta | Değişim Yok | Değişim Yok | Değişim Yok |
| Yüksek | Poz. Orta | Değişim Yok | Değişim Yok | Değişim Yok | Değişim Yok |
| Çok Yüksek | Değişim Yok | Değişim Yok | Değişim Yok | Değişim Yok | Değişim Yok |
| ABC Parameter | | | | | |
| GD,CS→ΔL | Koloni Boyutu | | | | |
| Genotipik Çeşitlilik(GD) | Düşük | Orta | Yüksek | | |
| Çok Düşük | Değişim Yok | Değişim Yok | Değişim Yok | | |
| Düşük | Değişim Yok | Değişim Yok | Değişim Yok | | |
| Orta | Değişim Yok | Değişim Yok | Poz. Orta | | |
| Yüksek | Değişim Yok | Poz. Orta | Poz. Yüksek | | |
| Çok Yüksek | Poz. Orta | Poz. Yüksek | Poz. Yüksek | | |

GA için iki ayrı bulanık çıkarsama sistemi çalışmaktadır. Bu sistemler sırası ile N ve Mo parametrelerinin değişim miktarının oranını temsilen ΔN ve ΔMo çıktılarını vermektedir. Bu iki değişkende aşağıda ki şekilde tanımlanmıştır.

ΔN = Popülasyon değişim oranı

$\Delta N \in [0.5,1,5] == [Neg.Yüksek, Poz.Yüksek]$

ΔMo = Mutasyon olasılığı değişim oranı

$\Delta Mo \in [0.5,1,5] == [Neg.Yüksek, Poz.Yüksek]$

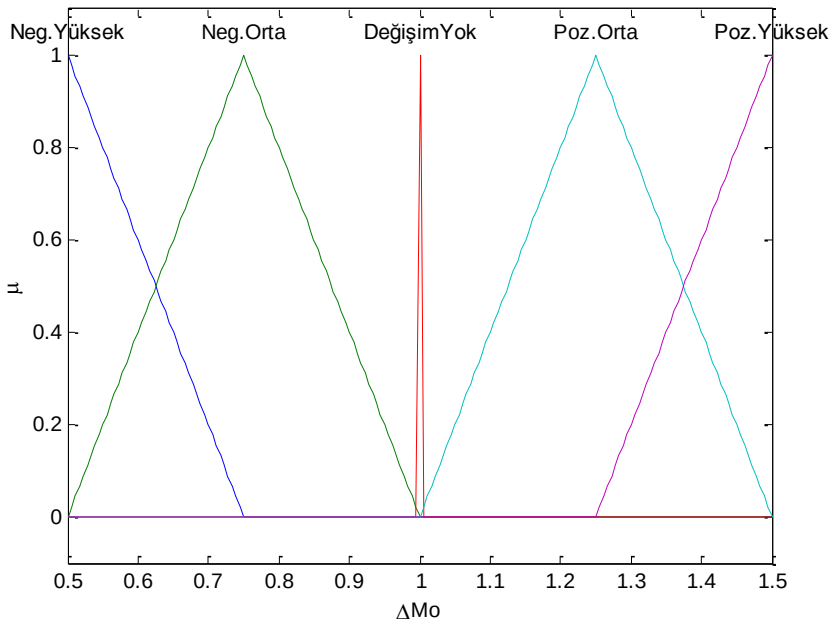
Bu değişkenlere ait üyelik fonksiyonları sırası ile Şekil 5.13 ve Şekil 5.12'de verilmiştir.

ABC'nin yalnızca L kontrol parametresine müdahele etmesi için tek bir bulanık çıkarsama sistemi vardır. Çıktı olarak ΔL parametresini vermektedir. Bu parametre;

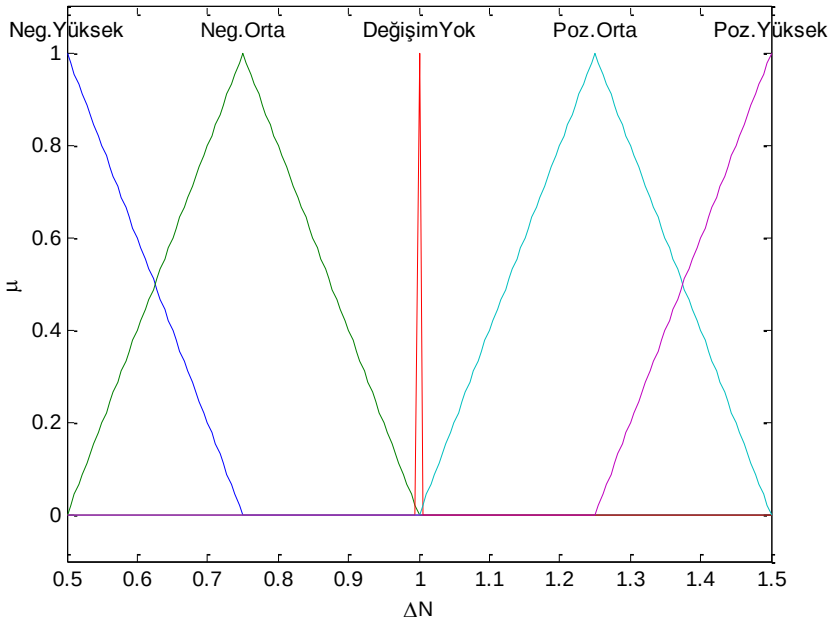
ΔL = Limit değişim oranı

$\Delta L \in [0.5,1,5] == [Neg.Yüksek, Poz.Yüksek]$

Şeklinde tanımlıdır. ΔL'ye ait üyelik fonksiyonları Şekil 5.14'de gösterildiği gibidir.




**Şekil 5.12.** Popülasyon değişim oranı değişkenine ait üyelik fonksiyonları




**Şekil 5.13.** Mutasyon olasılığı değişim oranı değişkenine ait üyelik fonksiyonları

Herhangi bir üst sezgisel için kontrol parametreleri ayarlanması yapılsa dahi, belirli bir zaman sonunda çözüm uzayının keşfedilmesi konusunda umut vaat edici davranmayabilir. Bu nokta çözüm uzayının keşfedilen kısmı ve üst sezgiselin anki ve geçmiş davranışları ile ilgili bazı hesaplamalar yapılır. Tasarlanan sistem için birden çok farklı parametre hesabı yapılır. Hesaplanacak parametrelerden biri SA için kullanılacak olan, normalleştirilmiş anlık sıcaklık (NAS) aşağıdaki gibi tanımlanır.

NAS= Normalleştirilmiş Anlık Sıcaklık

$$NAS = (Anlık\ Sıcaklık/Başlangıç\ Sıcaklık)$$

$$NAS \in [0,1] == [Neg.Yüksek, Poz.Yüksek]$$

Bu değişkene ait üyelik fonksiyonları Şekil 5.12 gibidir. Ayrıca SA için kullanılacak bir diğer parametre, gelişme göstermeyen amaç fonksiyonu değerlerinin normalleştirilmiş sıcaklık uzunluğudur (GADNSU). GASDNSU;

GASDNSU= Gelişme Göstermeyen Amaç Fonksiyonu Değerlerinin Normalleştirilmiş Sıcaklık Uzunluğu

$$\text{GASDNSU} = ((Anlık\ Sıcaklık - Son\ Gelişme\ Gösteren\ İterasyon\ Sıcaklığı)/Anlık\ Sıcaklık)$$

$$\text{GASDNSU} \in [0,1] == [Neg.Yüksek, Poz.Yüksek]$$

şeklinde tanımlanmıştır. GASDNSU ait üyelik fonksiyonları ise Şekil 5.13'de verilmiştir. Bahsi geçen iki parametrede SA' nin yeterliliği konusunda karar verilme aşamasında kullanılacaktır. ABC ve GA' algoritmaları için ise daha önce bahsedilen AFYD ve GÇ parametreleri dışında gelişme göstermeyen amaç fonksiyonu değerlerinin normalleştirilmiş uzunluğu (GADNU) kullanılacaktır. Bu parametre;

GASDNU= Gelişme Göstermeyen Amaç Fonksiyonu Değerlerinin Normalleştirilmiş Uzunluğu

$$\text{GASDNU} = ((Tamamlanan\ iterasyon\ sayısı - Son\ Gelişme\ Gösteren\ İterasyon\ Değeri)/Tamamlanan\ iterasyon\ sayısı)$$

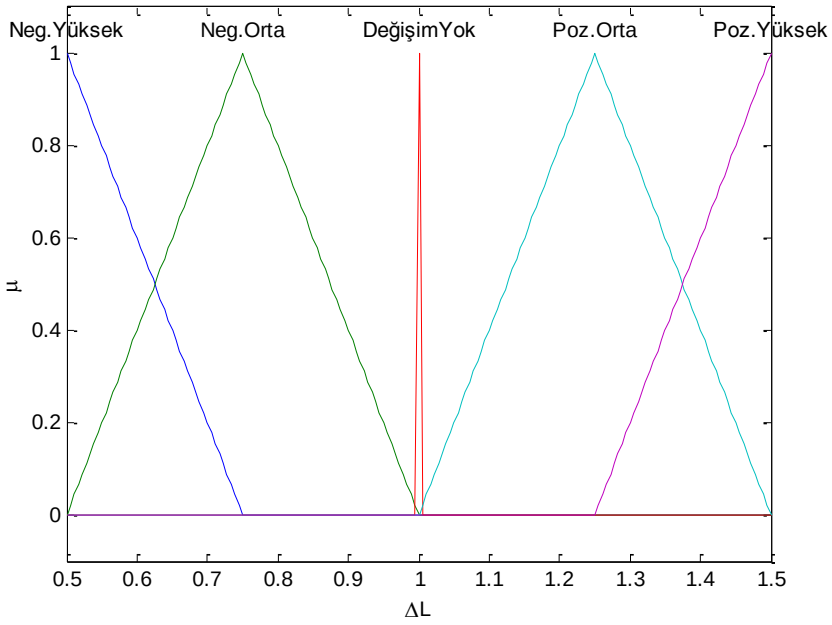
$$\text{GASDNU} \in [0,1] == [Neg.Yüksek, Poz.Yüksek]$$

biçiminde tanımlanmış olup; bu değişkene ait üyelik fonksiyonları Şekil 14'de verildiği gibidir.

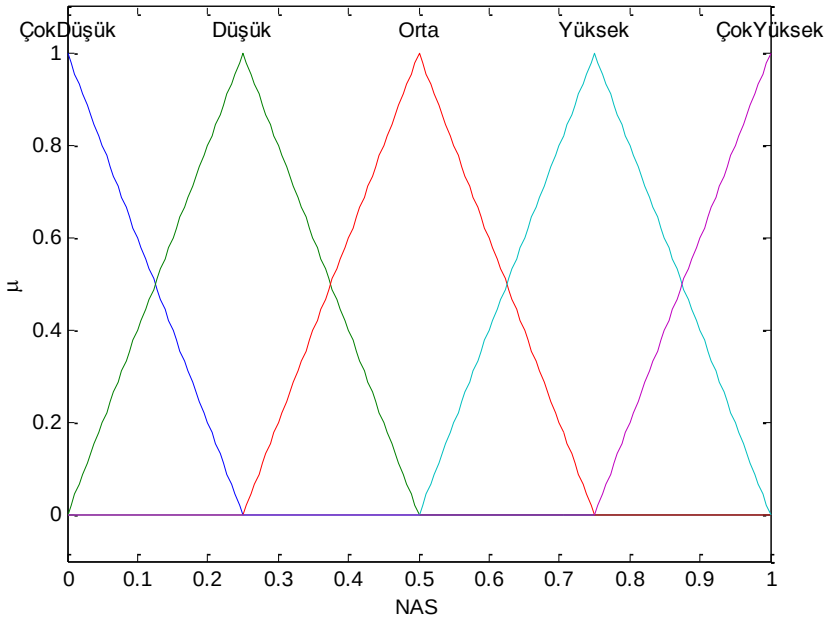
Tanımlanan parametreler her iterasyonda hesaplandıktan sonra aşağıdaki Çizelge 5.5'daki tabloda gösterilen kurallar yardımıyla, kullanılan üst sezgiselin bırakılıp bırakılmayacağına karar verilir.

**Çizelge 5.5.** Üst sezgisel bırakma kuraları

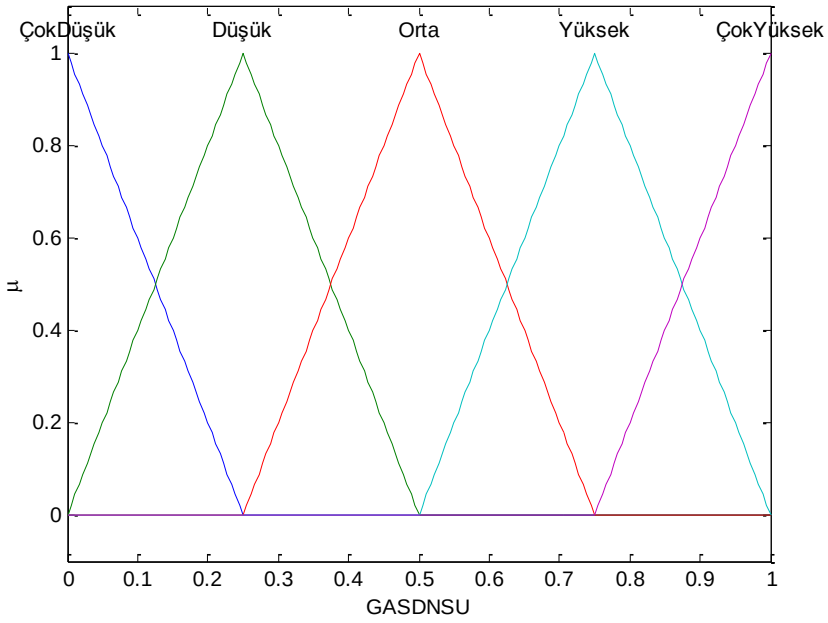
| SA Bırakma Kuralları | | | | | |
|---|---|---|---|---|---|
| | Normalleştirilmiş Anlık Sıcaklık (NAS) | | | | |
| Gelişme Göstermeyen Amaç Fonksiyonu Değerlerinin Normalleştirilmiş Sıcaklık Uzunluğu (GASDNSU ) | Çok Düşük | Düşük | Orta | Yüksek | Çok Yüksek |
| Çok Düşük | Bırakma | Bırakma | Bırakma | Bırakma | Bırak |
| Düşük | Bırakma | Bırakma | Bırakma | Bırak | Bırak |
| Orta | Bırakma | Bırakma | Bırak | Bırak | Bırak |
| Yüksek | Bırakma | Bırak | Bırak | Bırak | Bırak |
| Çok Yüksek | Bırak | Bırak | Bırak | Bırak | Bırak |
| GA Bırakma Kuralları | | | | | |
| | Genotipik Çeşitlilik(GD) | | | | |
| Gelişme Göstermeyen Amaç Fonksiyonu Değerlerinin Normalleştirilmiş Uzunluğu (GASDNU) | Çok Düşük | Düşük | Orta | Yüksek | Çok Yüksek |
| Çok Düşük | Bırakma | Bırakma | Bırakma | Bırakma | Bırakma |
| Düşük | Bırakma | Bırakma | Bırakma | Bırakma | Bırak |
| Orta | Bırakma | Bırakma | Bırakma | Bırak | Bırak |
| Yüksek | Bırakma | Bırakma | Bırak | Bırak | Bırak |
| Çok Yüksek | Bırakma | Bırak | Bırak | Bırak | Bırak |
| ABC Bırakma Kuralları | | | | | |
| | Amaç Fonksiyonu Yüzde Değerlendirmesi (AFYD | | | | |
| Gelişme Göstermeyen Amaç Fonksiyonu Değerlerinin Normalleştirilmiş Uzunluğu (GASDNU) | Çok Düşük | Düşük | Orta | Yüksek | Çok Yüksek |
| Çok Düşük | Bırakma | Bırakma | Bırakma | Bırakma | Bırakma |
| Düşük | Bırakma | Bırakma | Bırakma | Bırakma | Bırak |
| Orta | Bırakma | Bırakma | Bırakma | Bırak | Bırak |
| Yüksek | Bırakma | Bırakma | Bırak | Bırak | Bırak |
| Çok Yüksek | Bırakma | Bırak | Bırak | Bırak | Bırak |



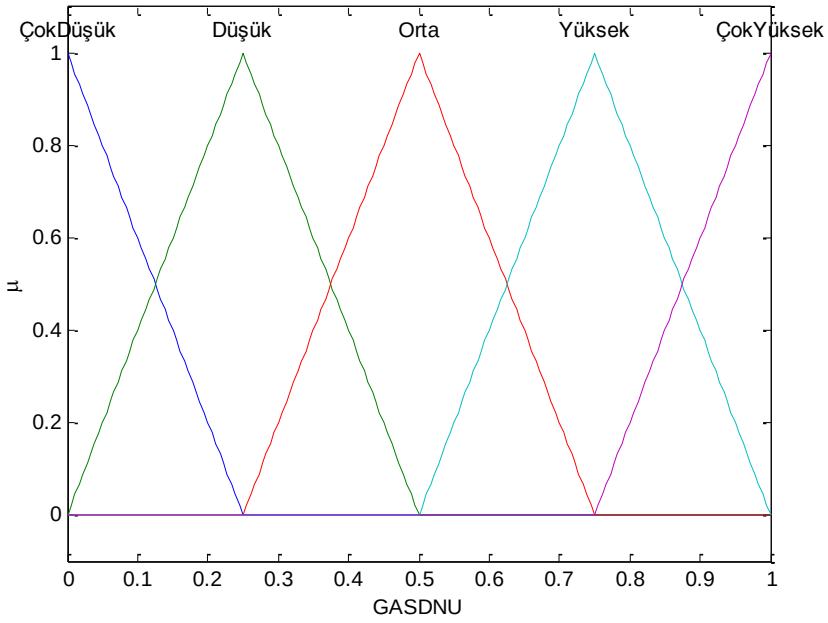

**Şekil 5.14.** Limit değişim oranı değişkenine ait üyelik fonksiyonları




**Şekil 5.15.** Normalleştirilmiş anlık sıcaklık değişkenine ait üyelik fonksiyonları




**Şekil 5.16** GASDNSU değişkenine ait üyelik fonksiyonu




**Şekil 5.17.** GASDNU değişkenine ait üyelik fonksiyonu

İlgilenilen üst sezgiselin bırakılmasına karar verilmesi yanında karar verilmesi gereken yeni bir durumu da getirir. Bu durumla başa çıkabilmek için; gerekli parametreleri önceden tanımlanmış, yeni bir dizi kural ve bu kuralları kullanan bulanık çıkarsama sistemleri vardır. Bırakılan üst sezgisele göre seçilecek yeni üst sezgisele karar veren kurallar Çizelge 5.6 ‘deki gibidir.

**Çizelge 5.6.** Bırakılan üst sezgisel yerine seçilecek üst sezgisel kuralları

| SA' dan geçilecek üst sezgisel | | |
|---|---|---|
| | Boyut | |
| Korelasyon | Küçük | Büyük |
| Korelasyonsuz | ABC | GA |
| Zayıf Korelasyonlu | ABC | GA |
| Korelasyonlu | GA | GA |
| GA' dan geçilecek üst sezgisel | | |
| | Boyut | |
| Genotipik Çeşitlilik(GD) | Küçük | Büyük |
| Çok Düşük | SA | SA |
| Düşük | SA | SA |
| Orta | ABC | SA |
| Yüksek | ABC | ABC |
| Çok Yüksek | ABC | ABC |
| ABC' dan geçilecek üst sezgisel | | |
| | Boyut | |
| Korelasyon | Küçük | Büyük |
| Korelasyonsuz | SA | SA |
| Zayıf Korelasyonlu | SA | SA |
| Korelasyonlu | GA | SA |

Buraya kadar verilen tüm kurallar ve anlatılan süreç her iterasyonda tekrarlanarak sistemin kontrolü sağlanır. Ancak sistemin aramayı ne zaman sonlandırılacağına da karar verilmesi gerekmektedir. Bu kararı vermek için kullanılacak kural tablosu Çizelge 5.7'da verilmiştir. Bu durum için daha önce tanımlanmış GASDNU parametresi dışında gelişme göstermeyen amaç fonksiyonu değerlerinin normalleştirilmiş zaman uzunluğu (GASDNZU) hesaplanması gereklidir. GASDNZU aşağıda ki şekilde tanımlanmıştır.

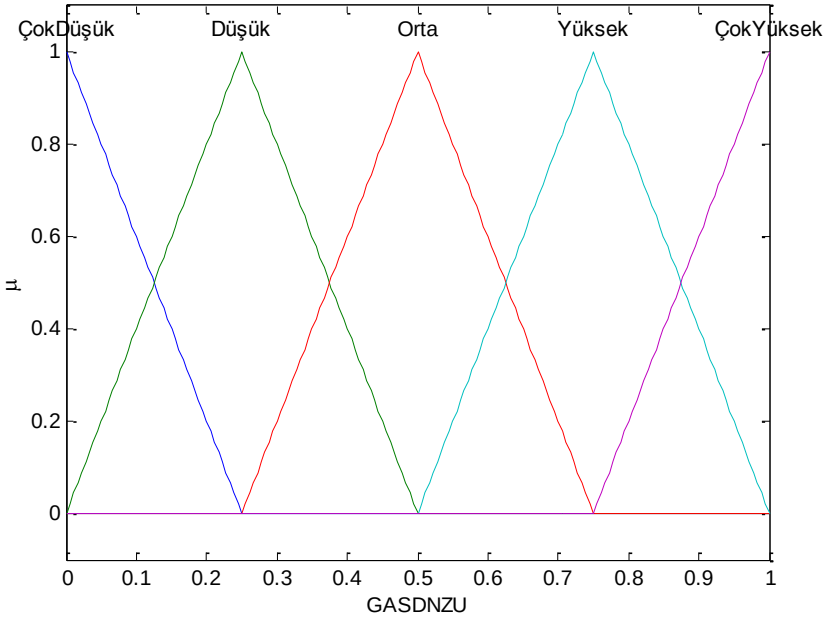
$$\text{GASDNZU} = ((Anlık\ Zaman - Son\ Gelişme\ Gösteren\ İterasyon\ Zamanı)/Anlık\ Zaman)$$

$$\text{GASDNZU} \in [0,1] == [Neg.Yüksek, Poz.Yüksek]$$

Ve bu değişkene ait üyelik fonksiyonları Şekil 5.18' daki gibidir.

**Çizelge 5.7.** Durdurma kuralları

| Durdurma Kuraları | | | | | |
|---|---|---|---|---|---|
| | Gelişme Göstermeyen Amaç Fonksiyonu Değerlerinin Normalleştirilmiş Zaman Uzunluğu | | | | |
| Gelişme Göstermeyen Amaç Fonksiyonu Değerlerinin Normalleştirilmiş Uzunluğu | Çok Düşük | Düşük | Orta | Yüksek | Çok Yüksek |
| Çok Düşük | Devam | Devam | Devam | Devam | Dur |
| Düşük | Devam | Devam | Devam | Dur | Dur |
| Orta | Devam | Devam | Dur | Dur | Dur |
| Yüksek | Devam | Dur | Dur | Dur | Dur |
| Çok Yüksek | Dur | Dur | Dur | Dur | Dur |




**Şekil 5.18.** GASDNZU değişkenine ait üyelik fonksiyonu

## 5.3. Hesaplama Denemeleri

Önerilen sistemin hesaplama denemeleri için toplam eşya sayısı $N$, eşya ağırlığı ve faydası için açıklık $R$ ve fayda ile ağırlık arası korelasyon tipi olmak üzere 3 parametreye göre 30 adet problem rasgele olarak üretilmiştir. Eşya sayısı 100 ve korelasyonu 0,03543872 olan problem üzerinden sistem içinde kullanılan üst sezgisellerin performansları ve önerilen sistemin performansları Çizelge 5.8,

Çizelge 5.9, Çizelge 5.10 ve Çizelge 5.11 verilmiştir. Bu probleme ait gerçek en iyi sonuç dinamik programlama kullanılarak elde edilmiş ve değerlendirme bunu baz alarak yapılmıştır. Performans değerlendirmelerinde ki süreler saniye bazın da ele alınmıştır. Çizelge 5.8 SA' algoritmasına ait performans değerlendirmesi verilmiştir.

**Çizelge 5.8.** SA için hesaplama denemeleri

| Eşya Sayısı | Korelasyon | Determinasyon | T | SF | Saniye(t) | f |
|---|---|---|---|---|---|---|
| 100 | 0,03543872 | 0,001255903 | 10 | 0,9 | 0 | 41479 |
| 100 | 0,03543872 | 0,001255903 | 10 | 0,95 | 0 | 41400 |
| 100 | 0,03543872 | 0,001255903 | 10 | 0,99 | 0 | 41494 |
| 100 | 0,03543872 | 0,001255903 | 100 | 0,9 | 0 | 41292 |
| 100 | 0,03543872 | 0,001255903 | 100 | 0,95 | 0 | 40998 |
| 100 | 0,03543872 | 0,001255903 | 100 | 0,99 | 0 | 41494 |
| 100 | 0,03543872 | 0,001255903 | 1000 | 0,9 | 0 | 41078 |
| 100 | 0,03543872 | 0,001255903 | 1000 | 0,95 | 0 | 41410 |
| 100 | 0,03543872 | 0,001255903 | 1000 | 0,99 | 0 | 41439 |

Çizelge 5.8 'da görüldüğü üzere SA algoritması farklı kontrol parametreleri durumlarında her zaman 1 saniyenin altında çalışmıştır. Ayrıca yapılan 9 denemenin 2 tanesinde gerçek en iyi sonucu yakalamıştır. Ürettiği sonuçların ortalaması 41342,67 olarak hesaplanmıştır. Sonuç olarak başarı yüzdesi %22,22 olarak söylenebilir.

GA'ya ait sonuçlar ise Çizelge 5.19 'da verildiği gibidir. Burada, farklı kontrol parametreleri ile 36 deneme yapılmıştır. Bu denemelerin 6 tanesinde süre 1 saniye üzerinde çıkmıştır. Diğerlerinde 1 saniyenin altındadır. Toplamda yapılan 36 denemenin yalnızca 3 tanesinde gerçek en iyi değer elde edilmiştir. GA sonuçlarının ortalaması 41089,25 olarak belirlenmiştir. Bu durumda, GA için %8,33'lük bir başarıdan söz edilebilir.

**Çizelge 5.9.** GA için hesaplama denemeleri

| Eşya Sayısı | Korelasyon | Determinasyon | N | İterasyon | ÇO | MO | Saniye(t) | f |
|---|---|---|---|---|---|---|---|---|
| 100 | 0,035438719 | 0,001255903 | 100 | 100 | 1 | 0,05 | 0 | 40618 |
| 100 | 0,035438719 | 0,001255903 | 100 | 100 | 1 | 0,1 | 0 | 40813 |
| 100 | 0,035438719 | 0,001255903 | 100 | 100 | 2 | 0,05 | 0 | 41154 |
| 100 | 0,035438719 | 0,001255903 | 100 | 100 | 2 | 0,1 | 0 | 40788 |
| 100 | 0,035438719 | 0,001255903 | 100 | 100 | 3 | 0,05 | 0 | 39820 |
| 100 | 0,035438719 | 0,001255903 | 100 | 100 | 3 | 0,1 | 0 | 40344 |
| 100 | 0,035438719 | 0,001255903 | 100 | 200 | 1 | 0,05 | 0 | 41245 |
| 100 | 0,035438719 | 0,001255903 | 100 | 200 | 1 | 0,1 | 0 | 41138 |
| 100 | 0,035438719 | 0,001255903 | 100 | 200 | 2 | 0,05 | 0 | 41295 |
| 100 | 0,035438719 | 0,001255903 | 100 | 200 | 2 | 0,1 | 0 | 41121 |
| 100 | 0,035438719 | 0,001255903 | 100 | 200 | 3 | 0,05 | 0 | 41021 |
| 100 | 0,035438719 | 0,001255903 | 100 | 200 | 3 | 0,1 | 0 | 40999 |
| 100 | 0,035438719 | 0,001255903 | 200 | 100 | 1 | 0,05 | 0 | 41325 |
| 100 | 0,035438719 | 0,001255903 | 200 | 100 | 1 | 0,1 | 0 | 41155 |
| 100 | 0,035438719 | 0,001255903 | 200 | 100 | 2 | 0,05 | 0 | 40751 |
| 100 | 0,035438719 | 0,001255903 | 200 | 100 | 2 | 0,1 | 0 | 41015 |
| 100 | 0,035438719 | 0,001255903 | 200 | 100 | 3 | 0,05 | 0 | 40396 |
| 100 | 0,035438719 | 0,001255903 | 200 | 100 | 3 | 0,1 | 0 | 40681 |
| 100 | 0,035438719 | 0,001255903 | 200 | 200 | 1 | 0,05 | 0 | 41494 |
| 100 | 0,035438719 | 0,001255903 | 200 | 200 | 1 | 0,1 | 0 | 41141 |
| 100 | 0,035438719 | 0,001255903 | 200 | 200 | 2 | 0,05 | 0 | 41407 |
| 100 | 0,035438719 | 0,001255903 | 200 | 200 | 2 | 0,1 | 0 | 41329 |
| 100 | 0,035438719 | 0,001255903 | 200 | 200 | 3 | 0,05 | 0 | 41312 |
| 100 | 0,035438719 | 0,001255903 | 200 | 200 | 3 | 0,1 | 0 | 41393 |
| 100 | 0,035438719 | 0,001255903 | 300 | 100 | 1 | 0,05 | 0 | 41261 |
| 100 | 0,035438719 | 0,001255903 | 300 | 100 | 1 | 0,1 | 0 | 41144 |
| 100 | 0,035438719 | 0,001255903 | 300 | 100 | 2 | 0,05 | 0 | 40844 |
| 100 | 0,035438719 | 0,001255903 | 300 | 100 | 2 | 0,1 | 0 | 41080 |
| 100 | 0,035438719 | 0,001255903 | 300 | 100 | 3 | 0,05 | 0 | 41119 |
| 100 | 0,035438719 | 0,001255903 | 300 | 100 | 3 | 0,1 | 0 | 41476 |
| 100 | 0,035438719 | 0,001255903 | 300 | 200 | 1 | 0,05 | 1 | 41479 |
| 100 | 0,035438719 | 0,001255903 | 300 | 200 | 1 | 0,1 | 1 | 41404 |
| 100 | 0,035438719 | 0,001255903 | 300 | 200 | 2 | 0,05 | 1 | 41411 |
| 100 | 0,035438719 | 0,001255903 | 300 | 200 | 2 | 0,1 | 1 | 41252 |
| 100 | 0,035438719 | 0,001255903 | 300 | 200 | 3 | 0,05 | 1 | 41494 |
| 100 | 0,035438719 | 0,001255903 | 300 | 200 | 3 | 0,1 | 1 | 41494 |

ABC' algoritmasının sonuçları Çizelge 5.10 'da verilmiştir. Bu sonuçlara göre, 9 farklı denemede elde edilen amaç fonksiyonu ortalaması 41168,11 olarak tespit edilmiştir. Bu denemelerin süreleri birinde 2 saniye üzerinde iken üçünde 1 saniye üzerindedir. Diğer denemelerde süre 1 saniyenin altında kalmıştır. ABC' algoritması denemelerin yalnızca 1'inde gerçek en iyi sonuca erişebilmiştir. Bu durumda başarı yüzdesi %11,1 olarak değerlendirilebilir.

**Çizelge 5.10.** ABC için hesaplama denemeleri

| Eşya Sayısı | Korelasyon | Determinasyon | Popülasyon | İterasyon | Saniye(t) | f |
|---|---|---|---|---|---|---|
| 100 | 0,03544 | 0,00126 | 250 | 50 | 0 | 40750 |
| 100 | 0,03544 | 0,00126 | 250 | 100 | 0 | 41273 |
| 100 | 0,03544 | 0,00126 | 250 | 150 | 0 | 41410 |
| 100 | 0,03544 | 0,00126 | 500 | 50 | 0 | 40754 |
| 100 | 0,03544 | 0,00126 | 500 | 100 | 0 | 41340 |
| 100 | 0,03544 | 0,00126 | 500 | 150 | 1 | 41479 |
| 100 | 0,03544 | 0,00126 | 1000 | 50 | 1 | 40673 |
| 100 | 0,03544 | 0,00126 | 1000 | 100 | 1 | 41340 |
| 100 | 0,03544 | 0,00126 | 1000 | 150 | 2 | 41494 |

**Çizelge 5.11.** Önerilen sistem için hesaplama denemeleri

| Eşya Sayısı | Korelasyon | Determinasyon | Saniye(t) | f |
|---|---|---|---|---|
| 100 | 0,03543872 | 0,0012559 | 1 | 41494 |
| 100 | 0,03543872 | 0,0012559 | 1 | 41494 |
| 100 | 0,03543872 | 0,0012559 | 1 | 41439 |
| 100 | 0,03543872 | 0,0012559 | 1 | 41494 |
| 100 | 0,03543872 | 0,0012559 | 1 | 41439 |
| 100 | 0,03543872 | 0,0012559 | 1 | 41494 |
| 100 | 0,03543872 | 0,0012559 | 1 | 41494 |
| 100 | 0,03543872 | 0,0012559 | 1 | 41494 |
| 100 | 0,03543872 | 0,0012559 | 1 | 41494 |

**Çizelge 5.12.** Ele alınan tüm çözüm yöntemleri elde edilen en iyi sonuçlar

| | | GA | | ABC | | SA | | MİS | |
|---|---|---|---|---|---|---|---|---|---|
| N | Determinasyon | f ortalama | f en iyi | f ortalama | f en iyi | f ortlaama | f en iyi | f ortalama | f en iyi |
| 100 | 0,001255903 | 41077,97 | 41494 | 41168,11 | 41494 | 41342,67 | 41494 | 41574 | 41494 |
| 100 | 0,457151127 | 46101,58 | 46251 | 46203,56 | 46251 | 46228,11 | 46251 | 46251 | 46251 |
| 100 | 0,996304339 | 9511,722 | 9664 | 9455,889 | 9652 | 9396,222 | 9564 | 9583,8 | 9664 |
| 100 | 0,002426656 | 188794,9 | 191977 | 189161,1 | 191977 | 191107,7 | 191886 | 191730,8 | 191886 |
| 100 | 0,42782146 | 217549,1 | 221371 | 219203 | 222512 | 220287,6 | 222836 | 222105,4 | 222512 |
| 100 | 0,998454524 | 542922,2 | 543878 | 542228,9 | 543453 | 540486,1 | 541976 | 543889 | 544044 |
| 200 | 0,002247565 | 62768,39 | 65589 | 63023,11 | 64917 | 65848,67 | 66007 | 65989,9 | 66029 |
| 200 | 0,495139208 | 55843,25 | 57676 | 56030,89 | 57877 | 58662,44 | 58947 | 58898,1 | 58975 |
| 200 | 0,996518239 | 10374,86 | 10779 | 10144,78 | 10571 | 10367,33 | 10679 | 10624,5 | 10779 |
| 200 | 0,005720816 | 965880,9 | 973437 | 965018,7 | 972631 | 972925,3 | 973785 | 973608,6 | 973950 |
| 200 | 0,454333083 | 914890,5 | 929173 | 917175 | 928210 | 932136,6 | 934042 | 934086,3 | 934871 |
| 200 | 0,996557686 | 156331,1 | 160788 | 153119,7 | 157202 | 157894,8 | 160794 | 160791,2 | 162794 |
| 500 | 0,000162761 | 230878,7 | 235014 | 231546,8 | 234811 | 236282,6 | 236490 | 236397,3 | 236486 |
| 500 | 0,463601547 | 43249,36 | 48502 | 42440,22 | 46588 | 51049,44 | 51895 | 51842,3 | 52043 |
| 500 | 0,997548676 | 246636,3 | 247740 | 245669,2 | 246108 | 247069,1 | 248012 | 247396,7 | 247911 |
| 500 | 0,000007099 | 2171315 | 2267568 | 2181327 | 2260962 | 2333540 | 2337929 | 2336078 | 2337875 |
| 500 | 0,542784394 | 2171567 | 2247271 | 2174920 | 2224679 | 2291158 | 2297446 | 2296211 | 2297575 |
| 500 | 0,996539199 | 2028279 | 2046811 | 2014705 | 2026555 | 2042769 | 2052010 | 2047309 | 2051858 |
| 1000 | 0,002567888 | 305259,5 | 336566 | 304588,1 | 328017 | 379921,3 | 380464 | 380176,8 | 380515 |
| 1000 | 0,514193087 | 36699,11 | 42723 | 35233,44 | 40230 | 48437,56 | 50312 | 50105,6 | 50613 |
| 1000 | 0,996479208 | 422381,4 | 426182 | 420188,7 | 422214 | 427853,6 | 429390 | 429530,8 | 430209 |
| 1000 | 0,001664271 | 1350323 | 1581395 | 1342561 | 1556148 | 1993371 | 2009619 | 2009686 | 2012786 |
| 1000 | 0,467250377 | 3262484 | 3462250 | 3256287 | 3391555 | 3817998 | 3831571 | 3830052 | 3831853 |
| 1000 | 0,997123456 | 2724687 | 2775765 | 2705535 | 2744293 | 2815624 | 2831398 | 2827135 | 2833582 |
| 2000 | 0,000839339 | 795955,9 | 843127 | 792721,8 | 823807 | 928772,2 | 929886 | 929520,2 | 930082 |
| 2000 | 0,50341555 | 480229,1 | 511710 | 473723,8 | 494568 | 609475,8 | 611621 | 611628,9 | 612321 |
| 2000 | 0,997140334 | 431586,3 | 439206 | 428300,4 | 433141 | 458294,8 | 461697 | 461496,9 | 461897 |
| 2000 | 0,000875049 | 5836866 | 6407324 | 5764306 | 6148891 | 7857859 | 7870296 | 7869609 | 7871492 |
| 2000 | 0,493208769 | 8620556 | 8880226 | 8573426 | 8758822 | 9575391 | 9602106 | 9601368 | 9603054 |
| 2000 | 0,997052377 | 5796941 | 5875847 | 5763562 | 5809825 | 6040183 | 6067498 | 6069058 | 6073441 |

Aynı problem özelinde tasarlanan sistem ele alındığında sonuçlar Çizelge 5.11 ‘daki gibidir. Çizelge 5.11’ daki sonuçlara bakıldığı zaman 9 deneme gerçekleştirildiği ve bu denemelerin hepsinde sürenin 1 saniyenin üzerinde olduğu görülmektedir. Ayrıca 9 ayrı denemenin 7’sinde yani yüzde %77,7’sinde sistemin en iyi sonucu elde ettiği görülmektedir. Sistemin elde ettiği sonuçların ortalaması 41582,89 olarak belirlenmiştir. Bu durumda her bir çözüm yaklaşımı ele alındığında, önerilen sistemin hem amaç fonksiyonları ortalaması hem de başarı yüzdesine göre en iyi yaklaşım olduğu görülmektedir. Çalışma süreleri bakımında da belirgin bir fark görülememektedir.

Çizelge 5.12 ‘da ise hem sistemin hem de diğer algoritmaların 30 probleme ’de verdikleri en iyi cevaplar ve ortalama cevaplar bazında karşılaştırması yer almaktadır. Çizelge 5.12 ki sonuçlarda, en iyi sonuçlar bilinen en iyi sonuçları temsil etmektedir. Sonuçlar incelendiğinde GA’nin 4, ABC’ nın 3, SA’ nin 7 ve önerilen sistemin 24 problem için bilinen en iyi sonuçları elde ettiği görülmektedir. Sonuçların bilinen en iyi olarak nitelendirilmesinin sebebi, belirli problemlerin boyutu ve çanta kapasiteleri nedeniyle makül zamanda kesin çözüme erişilememesidir. Ayrıca ortalama amaç fonksiyonuna göre sonuçlar değerlendirildiğinde önerilen sistemin tüm durumlarda en iyi ortalama değerlere ulaştığı görülmektedir.

# 6. SONUÇ

En yaygın optimizasyon çözüm yöntemlerinden biri olan üst sezgisellerin, hangi problem için hangi algoritma ve hangi kontrol parametreleri sorusu bu çalışmanın çıkış noktasıdır. Herhangi bir optimizasyon problemi için uygun algoritmayı ve kontrol parametrelerini belirlemek için konvansiyonel yaklaşım farklı durumlar için algoritmayı deneyerek en uygun olanına karar vermektir. Tez çalışmasının odak noktası burasıdır. Ayrıca tez çalışmasında birden farklı üst sezgiselin avantajlı yanlarını bir araya getirerek daha iyi bir arama yaklaşımı elde edilmeye çalışılmıştır.

Tez çalışmasında ilgilenilen problem 0-1 Sırt Çantası olmuştur. Bu problem için yapay veri setleri türetilmiştir. 0-1 Sırt Çantası, uygun üst sezgiseller GA, ABC ve SA ele alınmıştır. Bu 3 üst sezgiseli kullanan, bir işbirlikçi sistem tasarlanmıştır. Bu sistem karar aşamalarında bulanık mantık kullanmıştır.

Önerilen sistem ve bahsi geçen üst sezgiseller ayrı ayrı veri setleri üzerinde hesaplama denemeleri gerçekleştirmiştir. Bu denemeler sonucunda, önerilen sistemin hız açısından diğer sistemlerden bir farkının olmamasına karşın, çözüm uzayını çok daha etkin şekilde aradığı ortaya konmuştur. Deney sonuçları, sistem ortaya konarken ifade edilen beklentileri karşıladığını göstermiştir.

Gelecekte, önerilen sistem de pek çok iyileştirme yapılabileceği açıktır. Sisteme ait kurallar sezgisel olarak oluşturulmuş olsa da, bu kuralların sistematik bir yapıda oluşturulması mümkündür. Sistematik bir yaklaşımın verimi artırabileceği düşünülmektedir. Ayıraca bulanık mantık dışında da bir takım esnek hesaplama teknikleri ve öz uyarlamalı tekniklerde önerilen sisteme entegre edilebilir. Bu entegrasyonunda sistemi geliştireceği ön görülmektedir. Ayrıca önerilen sistem konusunda önceden yapılmış çok fazla çalışma olmaması da sistemin gelişmeye açık olduğu ve üzerinde çalışılacak alanların çokluğunun bir göstergesidir.

# KAYNAKLAR DİZİNİ

**Bäck T. and Schwefel H.P.,** 1993, An overview of evolutionary algorithms for parameter optimization, Evolutionary Computation 1:1–23 pp.

**Battiti R. and Tecchiolli G.,** 1994, The reactive tabu search, ORSA Journal on Computing 6:126–140 pp.

**Birattari M., Paquete L., Stützle T. and Varrentrapp K.,** 2001, Classification of Metaheuristics and Design of Experiments for the Analysis of Components, Technical Report AIDA-01-05, FG Intellektik, FB Informatik, Technische Universität Darmstadt, Darmstadt, Germany.

**Blickle T. and Thiele L.,** 1995, A comparison of selection schemes used in genetic algorithms, Evolutionary Computation 4:311–347 pp.

**Blum C. and Roli A.,** 2003, Metaheuristics in combinatorial optimization: overview and conceptual comparison, ACM Computing Surveys 35:268–308 pp.

**Bouusaid I., Lepagnot J. and Siarry P.,** 2010, A survey on optimization metaheuristics, Information Science 237:82-117 pp.

**Burke E. K., Kendall G., and Soubeiga E.,** 2003, A Tabu-Search Hyper-Heuristic for Timetabling and Rostering. Journal of Heuristics, 9(6):451-470 pp.

**Cadenes J. M., Garrido M. C., Hernandez L. D. and Munoz E.,** 2006, Towards a definition of a Data Mining process based on Fuzzy Sets for Cooperativo Metaheuristic systems, International Conference on Information Processing and Management of Uncertainty in Knowledge-Based Systems, Pars, 2825-2835 pp.

**Chelouah R. and Siarry P.,** 2000, Tabu search applied to global optimization, European Journal of Operational Research 123:256–270 pp.

**Cormen T.H., Leiserson C. E., Rivest R. L. and Stein C.,** 2009, Introduction to Algorithms (3rd ed.), MIT Press and McGraw-Hill, 1312 p.

**Cvijovic D. and Klinowski J.,** 1995, Tabu search: an approach to the multiple minima problem, Science 267:664–666 pp.

**Dantzig G.B.,** 1957, Discrete Variable Extremum Problems, Operations Research 5:266-277 pp.

## KAYNAKLAR DİZİNİ(devam)

**Denzinger J., Fuchs M. and Fuchs M.,** 1997, High Performance ATP Systems by Combining Several AI Methods, In Proceedings of the Fifteenth International Joint Conference on Artificial Intelligence, 102-107 pp.

**Dorigo M. and Blum C.,** 2005, Ant colony optimization theory: a survey, Theoretical Computer Science 344:243–278 pp.

**Dorigo M. and Maniezzo V.,** 1991, A. Colorni, Positive Feedback as a Search Strategy, Technical Report 91-016, Dipartimento di Elettronica, Politecnico di Milano, Milan, Italy.

**Dorigo M. and Maniezzo V.,** 1996, A. Colorni, The ant system: optimization by a colony of cooperating agents, IEEE Transactions on Systems, Man, and Cybernetics – Part B 26:29–41 pp.

**Dorigo M. and Thomas S.,** 2010, Ant colony optimization: overview and recent advances, Handbook of Metaheuristics, Gendreau M. and Potvin J.Y. (EDS.), International Series in Operations Research & Management Science, Springer, US, 227–263 pp.

**Dorigo M.,** 1992, Optimization, Learning and Natural Algorithms, Ph.D. Thesis, Politecnico di Milano, Italy.

**Dorigo M., Birattari M., Stützle T., Libre U. and Bruxelles D.,** 2006, A.F.D. Roosevelt, Ant colony optimization – artificial ants as a computational intelligence technique, IEEE Computational Intelligence Magazine 1:28–39 pp.

**Dubois, D. and Prade, H.,** 1980, Fuzzy Sets and Systems: Theory and AppHcations. Academic, New York.

**Eles P. And Peng Z.,** 2010, Heuristic Algorithms for Combinatorial Optimization Problems, http://www.ida.liu.se/~zebpe/heuristic/lectures/lec1.pdf (Erişim Tarihi: 17 Mayıs 2014)

**Fisher H. and Thompson G. L., 1961,** Probabilistic learning combinations of local job-shop scheduling rules, Factory Scheduling Conference, Carnegie Institute of Technology.

**Fisher H. and Thompson G. L., 1963,** Probabilistic learning combinations of local job-shop scheduling rules,Industrial Scheduling, Muth J.F. and Thompson G.L. (EDS.), Prentice-Hall, Inc, New Jersey, 225–251 pp.

## KAYNAKLAR DİZİNİ(devam)

**Glover F.,** 1977, Heuristics for integer programming using surrogate constraints, Decis. Sci. 8(1):156–166 pp.

**Glover F.,** 1986, Future paths for integer programming and links to artificial intelligence, Computers and Operations Research 13:533–549 pp.

**Glover F.,** 1994, Tabu search for nonlinear and parametric optimization (with links to genetic algorithms), Discrete Applied Mathematics 49:231– 255 pp.

**Goldberg D.E. and Deb K.,** 1991, A comparative analysis of selection schemes used in genetic algorithms, Foundations of Genetic Algorithms, Morgan Kaufman, 69-93 pp.

**Hirschberg D.S. and Wong C.K.,** 1976, A polynomial-time algorithm for the knapsack problem with two variables, Journal ACM 23:147-154 pp.

**Holland J.H.,** 1975, Adoptaion in Natural and Artificial Systems. Universty of Michigan Press.

**Kannan R.,** 1980, A polynomail algorithm fort he two-variable integer programming problem, Journal ACM 30:133-145 pp.

**Karaboga D.,** 2005, An Idea Based on Honey Bee Swarm for Numerical Optimization, Technical Report TR06, Erciyes University.

**Karaboğa D.,** 2011, Yapay Zeka Optimizasyon Algoritmaları, Nobel Yayın, 244s.

**Karp R.M.,** 1972, Reducibility among Combinatorial Problems, Complexity of Computer Computations, New York, 85–103 pp.

**Karr, C. L. and Gentry, E. J.,** 1993, Fuzzy control of pH using genetic algorithms, IEEE Trans. Fuzzy Syst. 1:46-53 pp.

**Kirkpatrick S., Gelatt C. and Vecchi M.,** 1983, Optimization by simulated annealing, Science 220:671–680 pp.

**Larsen P.M.,** 1980, Industrial applications of fuzzy logic control, Int. J. Man Mach. Studies 12(1):3 -10 pp.

**Lawler E.L., Lenstra J.K., Rinnooy Kan A.H.G. and Shmoys D.B.,** 1985, The Traveling Salesman Problem, John Wiley & Sons Ltd., Chichester 473 p.

**Lee, C. C,** 1990, Fuzzy logic in control systems: fuzzy logic controllers—parts I and II, IEEE Trans. Syst. Man. Cybernet. 20:404-435 pp.

## KAYNAKLAR DİZİNİ(devam)

**Lucic P. and Teodorovic D.,** 2003, Computing with bees: attacking complex transportation engineering problems, International Journal on Artificial Intelligence Tools 12:375–394 pp.

**Lueker G. S.,** 1975, Report No. 178, Computer Science Laboratory, Princeton.

**Luke S,** 2013, Essentials of Metaheuristics(Second Edition), Lulu, 242 p.

**Mamdani E.H,** 1976, Advances in the linguistic synthesis of fuzzy controllers, International Journal of Man-Machine Studies 8(6):669–678 pp.

**Martello S. and Toth P.,** 1990, Knapsack problems: algorithms and computer implementations, John Wiley & Sons, Inc., New York, 296 p.

**Martello S., Pisinger D. and Toth P.,** 1999, Dynamic programming and strong bounds for the 0-1 knapsack problem, Management Science 45(3):414-424 pp.

**Metropolis N., Rosenbluth A., Rosenbluth M., Teller A. and Teller E.,** 1953, Equation of state calculations by fast computing machines, Journal of Chemical Physics 21:1087–1090 pp.

**Mladenovic N. and Hansen P.,** 1997, Variable neighborhood search, Computers and Operations Research 24:1097–1100 pp.

**Mladenovic N.,** 1995, A variable neighborhood algorithm – a new metaheuristic for combinatorial optimization, in: Abstracts of Papers Presented at Optimization Days, Montréal, Canada, 112 p.

**Nauss R. M.,** 1976, An efficient algorithm for the 0-1 knapsack problems, Management Science 23:27-31 pp.

**Papis C. and Siettos C.,** 2005, Chapter 15. Fuzzy Reasoning, in Introductory Tutorials In Optimization and Search Methodologies, Burke E. and Kendall G.(EDS.), Springer-Verlag.

**Rice J. R.,** 1976, The algorithm selection problem, Advances in Computers 15:65-118 pp.

**Ross, T. J.,** 1995, Fuzzy Logic with Engineering Applications, McGraw-Hill, New York, 579 p.

**Russel S. J. and Norving P.,** 2010, Artificial Intelligence: A Modern Approach, Prentice Hall Inc, Upper Saddle River, 1132p.

## KAYNAKLAR DİZİNİ(devam)

**Sugeno, M. and Yasukawa, T.,** 1993, A fuzzy-logic-based approach to qualitative modelling IEEE Trans. Fuzzy Syst. 1:7-31 pp.

**X.S. Yang,** 2005, Engineering optimizations via nature-inspired virtual bee algorithms, Artificial Intelligence and Knowledge Engineering Applications: A Bioinspired Approach, First International Work-Conference on the Interplay Between Natural and Artificial Computation, Lecture Notes in Computer Science 3562, Springer, 317–323 pp.

**Yan, J., Ryan, M. and Power, J.,** 1994, Using Fuzzy Logic, Prentice Hall Inc., Englewood Cliffs, New Jersey, 256 p.

**Zadeh L. A.,** 1965, Fuzzy Sets, Information and Control 8:338-353 pp.

**Zimmermann, H. J.,** 1996, Fuzzy Set Theory and its Applications, 3rd edn, Kluwer, Dordrecht, 514 p.

# ÖZGEÇMİŞ

Barış Tekin Tezel, 23 Mayıs 1989 tarihinde İstanbul'da doğmuştur ve Türkiye Cumhuriyeti vatandaşıdır. Liseyi 2003–2007 yılları arasında Deniz Lisesi Komutanlığı'ında okumuştur. Haziran 2012'de Ege Üniversitesi İstatistik Bölümün'den bölüm ikincisi ve yüksek onur öğrencisi olarak mezun olmuştur. Eylül 2012'de Ege Üniversitesi Fen Bilimleri Enstitüsü İstatistik Anabilim Dalı'nda yüksek lisans eğitimine başlamıştır. Ocak 2014'de Dokuz Eylül Üniversitesi Bilgisayar Bilimleri Bölümü'nde Araştırma Görevlisi olarak göreve başlamıştır.

# EKLER

Ek 1 Form1 sınıfı

Ek 2 ABC sınıfının fonksiyon deklarasyonları

Ek 3 GA sınıfının fonksiyon deklarasyonları

Ek 4 SA sınıfının fonksiyon deklarasyonları

Ek 5 Rasgele sayı üretim sınıfı

Ek 6 Bulanık küme sınıfı

**Ek 2 Form1 Sınıfı "Form1.cs"**

```
public partial class Form1 : Form
    {
        public long nanosecPerTick
        {
            get{ return (1000L * 1000L * 1000L) / Stopwatch.Frequency;}
        }
        public static double gbestf;
        public static int sayac;
        public static int bestsayac;
        public static int sec;
        public static int olsec;
        public static Stopwatch time = new Stopwatch();
        public static double  best;
        public static int sonpop;
        public static Queue sira = new Queue();
        public static int algsay;
        StreamWriter writer ;
        public Form1()
        {

            InitializeComponent();
        }
        private void BaslatButton_Click(object sender, EventArgs e)
        {
            this.TransparencyKey = Form.DefaultBackColor;
            Application.DoEvents();
            time.Start();
            writer = new StreamWriter("sonuc.txt",true);
            algsay = 1;
            int esya = Convert.ToInt32(textBox1.Text);
            double limit = Convert.ToDouble(textBox2.Text);
            olsec = 0;
            best = 0;
            sonpop = 0;
            gbestf = 0;
            sayac = 1;
            bestsayac = 1;
            int pop;
            sira.Clear();
            double[] agirlik = new double[esya];
            double[] degerler = new double[esya];
            DiziDoldur(agirlik, degerler, esya);
            sec = basalgsec(agirlik, degerler, esya);
            Fuzzy f = new Fuzzy();
            SA s = new SA();
            GA g = new GA();
            ABC a = new ABC();
            byte[] gen = new byte[esya];
            byte[,] dizi = new byte[1000, esya];
            double[] ftnss = new double[1000];
            int[] fsay = new int[1000];
            double[] par;
            double mo = 0;
            do
            {
                switch (sec)
                {
                    case 1:
                        par = a.AbcParam(agirlik, degerler, esya);
```

**Ek 2 Form1 Sınıfı "Form1.cs"(devam)**

```
                        pop = (int)par[0];
                        if (olsec == 0)
                            a.PopOlAbc(dizi, agirlik, fsay, esya, pop,
limit);
                        else if (olsec == 2)
                        {
                            a.PopOlAbcFromGa(dizi, agirlik, fsay, esya,
pop, limit);
                        }
                        else if (olsec == 3)
                        {
                            a.PopOlAbcFromSa(gen, dizi, agirlik, fsay,
esya, pop, limit);
                        }
                        for (int i = 0; i < pop; i++)
                        {
                            double toplam = 0;
                            for (int k = 0; k < esya; k++)
                            {
                                toplam = toplam + ((double)dizi[i, k]) *
degerler[k];
                            }
                            ftnss[i] = toplam;
                        }
                        Sıralama(ftnss, dizi, 0, pop - 1,esya);
                        a.ABCİterasyon(dizi, ftnss, degerler, agirlik,
fsay, esya, pop, (int)par[1], limit);
                        break;
                    case 2:

                        par = g.GaParam(agirlik, degerler, esya);
                        pop = (int)par[1];
                        mo = par[0];
                        if (olsec == 0)
                        {
                            g.PopOlGa(dizi, agirlik, esya, pop, limit);
                        }
                        else if (olsec == 1)
                        {
                            g.PopOlGAFromABC(dizi, agirlik, esya, pop,
limit);
                        }
                        else if (olsec == 3)
                        {
                            g.PopOlGaFromSa(gen, dizi, agirlik, esya, pop,
limit);
                        }
                        for (int i = 0; i < pop; i++)
                        {
                            double toplam = 0;
                            for (int k = 0; k < esya; k++)
                            {
                                toplam = toplam + ((double)dizi[i, k]) *
degerler[k];
                            }
                            ftnss[i] = toplam;
                        }
                        Sıralama(ftnss, dizi, 0, pop - 1,esya);
```

**Ek 2 Form1 Sınıfı "Form1.cs"(devam)**

```
                        g.GenetikİterasyonGA(dizi, ftnss, degerler, agirlik,
esya, pop, mo, limit);
                        break;
                       case 3:
                        par = s.SaParam(agirlik, degerler, esya);
                        if (olsec == 0)
                            s.BasOlustur(gen, agirlik, esya, limit);
                        else
                            sacopy(gen, dizi, esya);
                        s.Sa(gen, agirlik, degerler, par[0], par[1], esya,
limit);
                        algsay++;
                        break;
                }
                double ha = 0;
                int haS = 0;
                object[] array = sira.ToArray();
                int aL = (array.Length > 4) ? array.Length - 3 : 0;
                for (int y = array.Length - 1; y > aL; y--)
                {
                    ha += Convert.ToDouble(array[y]);
                    haS++;
                }
                haS = (haS == 0) ? 1 : haS;
                ha = ha / haS;
                double elaps = (double)((time.ElapsedTicks)
/nanosecPerTick);
                double NTUF = (elaps-best) / elaps;
                double NUF = ((double)(sayac - bestsayac) / (double)(sayac
- ha));
                bool cevap = stop(NTUF, NUF);
                if (cevap == true) break;

            } while (true);
            time.Stop();
            writer.WriteLine(String.Format("N= {0} Kapasite= {1} Maks= {2}
Geçen Zaman={3}",esya,limit, gbestf, (double)(time.ElapsedTicks /
(double)(Stopwatch.Frequency))));
            time.Reset();
            writer.Close();
            this.AllowTransparency = false;
            Application.DoEvents();
        }
        private bool stop(double NTUF, double NUF)
        {
            double[,] initialAlg = new double[5, 5];
            Fuzzy f = new Fuzzy();
            double y = 0, l = 0;
            double max = -1;
            int imax = 0, jmax = 0;
            for (int i = 0; i < 5; i++)
                for (int j = 0; j < 5; j++)
                {
                    switch (i)
                    {
                        case 0: l = f.trimf(NUF, Fuzzy.NUF_vl); break;
                        case 1: l = f.trimf(NUF, Fuzzy.NUF_l); break;
                        case 2: l = f.trimf(NUF, Fuzzy.NUF_m); break;
                        case 3: l = f.trimf(NUF, Fuzzy.NUF_h); break;
```

**Ek 2 Form1 Sınıfı "Form1.cs"(devam)**

```
                        case 4: l = f.trimf(NUF, Fuzzy.NUF_vh); break;
                    }
                    switch (j)
                    {
                        case 0: y = f.trimf(NTUF, Fuzzy.NTUF_vl); break;
                        case 1: y = f.trimf(NTUF, Fuzzy.NTUF_l); break;
                        case 2: y = f.trimf(NTUF, Fuzzy.NTUF_m); break;
                        case 3: y = f.trimf(NTUF, Fuzzy.NTUF_h); break;
                        case 4: y = f.trimf(NTUF, Fuzzy.NTUF_vh); break;
                    }
                    initialAlg[i, j] = f.min(y, l);
                    if (max < initialAlg[i, j])
                    {
                        max = initialAlg[i, j];
                        imax = i;
                        jmax = j;
                    }
                }
            if ((imax+ jmax>5)||(imax==4)||(jmax==4)) return true;
            else return false;
        }
        private void sacopy(byte[] gen,byte[,] dizi,int esya)
        {
            for (int i = 0; i < esya; i++)
                gen[i] = dizi[0, i];
        }
        public int basalgsec(double[] agirlik,double[] degerler, int esya)
        {
                double[,] initialAlg = new double[3, 2];
                double x = detcof(agirlik, degerler, esya);
                Fuzzy f = new Fuzzy();
                double y = 0, l = 0;
                double max = -1;
                int imax = 0, jmax = 0;
                for (int i = 0; i < 3; i++)
                    for (int j = 0; j < 2; j++)
                    {
                        switch (i)
                        {
                            case 0: y = f.trimf(x, Fuzzy.cor_u); break;
                            case 1: y = f.trimf(x, Fuzzy.cor_w); break;
                            case 2: y = f.trimf(x, Fuzzy.cor_c); break;
                        }
                        switch (j)
                        {
                            case 0: l = f.trimf(esya, Fuzzy.p_size_s);
break;
                            case 1: l = f.trimf(esya, Fuzzy.p_size_l);
break;
                        }

                        initialAlg[i, j] = f.min(y, l);
                        if (max < initialAlg[i, j])
                        {
                            max = initialAlg[i, j];
                            imax = i;
                            jmax = j;
                        }
                    }
```

**Ek 2 Form1 Sınıfı "Form1.cs"(devam)**

```
                    if(imax==2 && jmax==0) return 2;
                    else  return 3;

        }
        public double detcof(double[] w,double[] p,int n)
        {
            double a = 0, b = 0, ab = 0, a2 = 0, b2 = 0;
            for (int m = 0; m < n; m++)
            {
                a += w[m];
                b += p[m];
                ab += (w[m] * p[m]);
                a2 += w[m] * w[m];
                b2 += p[m] * p[m];
            }
            double cor = (n * ab - a * b) / (Math.Sqrt(n * a2 - (a * a)) *
Math.Sqrt(n * b2 - (b * b)));
            return Math.Pow(cor, 2);
        }
        public void Sıralama(double[] fit, byte[,] dizi, int left, int
right,int esya)
        {
            int p = left, q = right, x = (p + (q - p) / 2);
            double m = fit[x];
            double big = 0;
            byte big1;
            do
            {
                if (p < fit.Length) while (fit[p] > m) p++;
                if (q > -1) while (fit[q] < m) q--;
                if (p > q) break;
                big = fit[p];
                fit[p] = fit[q];
                fit[q] = big;
                for (int i = 0; i <esya; i++)
                {
                    big1 = dizi[p, i];
                    dizi[p, i] = dizi[q, i];
                    dizi[q, i] = big1;
                }
            } while (p++ <= q--);
            if (left < q) Sıralama(fit, dizi, left, q,esya);
            if (p < right) Sıralama(fit, dizi, p, right,esya);
        }
        private void DiziDoldur(double[] agirlik, double[] degerler,int
esya)
        {
            string bos = "";
            bos = richTextBox3.Text;
            char[] ayr = new char[2];
            ayr[0] = ' ';
            ayr[1] = '\n';
            string[] cc = bos.Split(ayr);
            for(int i=0;i<esya;i++)
            {
                agirlik[i] = Convert.ToDouble(cc[i]);
            }
            bos = richTextBox2.Text;
            string[] cc1 = bos.Split(ayr);
```

**Ek 2 Form1 Sınıfı "Form1.cs"(devam)**

```
            for (int i = 0; i < esya; i++)
            {
                degerler[i] = Convert.ToDouble(cc1[i]);

            }
        }
        private void Form1_Load(object sender, EventArgs e)
        {
              writer = new StreamWriter("sonuc.txt");
              writer.Close();
              richTextBox2.WordWrap = false;
              richTextBox2.ScrollBars =
RichTextBoxScrollBars.ForcedHorizontal;
              richTextBox3.WordWrap = false;
              richTextBox3.ScrollBars =
RichTextBoxScrollBars.ForcedHorizontal;

        }
        private void button1_Click(object sender, EventArgs e)
        {
            this.Close();
        }

        private void button2_Click(object sender, EventArgs e)
        {
            textBox2.Text = "";
            richTextBox3.Text = "";
            richTextBox2.Text = "";
            String input = string.Empty;
            OpenFileDialog dialog = new OpenFileDialog();
            dialog.Filter = "txt dosyaları (*.txt)|*.txt|Tüm dosyalar
(*.*)|*.*";
            dialog.InitialDirectory = Application.StartupPath;
            dialog.Title = "Select a txt file";
            if (dialog.ShowDialog() == DialogResult.OK)
                input = dialog.FileName;
            if (input == String.Empty)
                return;
            string filePath = (/*Application.StartupPath +
*/input).ToString();
            string line;
            if (File.Exists(filePath))
            {
                StreamReader file = null;
                try
                {
                    file = new StreamReader(filePath);
                    line = file.ReadToEnd();
                    int r = 0, i,es=0;
                    string sayi = "";
                    char[] parcali = line.ToCharArray();
                    for (i = 0; parcali[i] != '\n'; i++)
                    {
                        sayi += parcali[i];
                    }
                    r = i;
                    textBox2.Text += sayi;
                    sayi = "";
```

**Ek 2 Form1 Sınıfı "Form1.cs"(devam)**

```
                    for (i = r; parcali[i] == '\n' || parcali[i] == '\r';
i++) r = i;
                    for (i = r + 1; parcali[i] != '\n'; i++)
                    {
                        sayi += parcali[i];
                    }
                    r = i;
                    richTextBox3.Text = sayi;
                    sayi = "";
                    for (i = r; parcali[i] == '\n' || parcali[i] == '\r';
i++) r = i;
                    for (i = r + 1; parcali[i] != '\n'; i++)
                    {
                        if (parcali[i] == ' ') es++;
                        sayi += parcali[i];

                    }
                    textBox1.Text = es.ToString();
                    richTextBox2.Text = sayi;

                }

                finally
                {
                    if (file != null)
                        file.Close();
                }
            }
        }
        private void button3_Click_1(object sender, EventArgs e)
        {
            String input = string.Empty;
            for (int o = 0; o < 5; o++)
            {
                int size = 0;
                switch (o)
                {
                    case 0: size = 100; break;
                    case 1: size = 200; break;
                    case 2: size = 500; break;
                    case 3: size = 1000; break;
                    case 4: size = 2000; break;

                }
                for (int j = 1000; j <= 10000; j *= 10)
                {
                    for (int l = 1; l < 4; l++)
                    {
                        textBox2.Text = "";
                        richTextBox3.Text = "";
                        richTextBox2.Text = "";
                        input = "N "+size.ToString()+" R "+j.ToString()+" T
"+l.ToString();
                        string filePath = (Application.StartupPath + "\\" +
input + ".txt").ToString();
                        string line;

                        if (File.Exists(filePath))
                        {
```

**Ek 2 Form1 Sınıfı "Form1.cs"(devam)**

```
                            StreamReader file = null;
                            try
                            {
                                file = new StreamReader(filePath);
                                line = file.ReadToEnd();
                                int r = 0, i, es = 0;
                                string sayi = "";
                                char[] parcali = line.ToCharArray();
                                for (i = 0; parcali[i] != '\n'; i++)
                                {
                                    sayi += parcali[i];
                                }
                                r = i;
                                textBox2.Text += sayi;
                                sayi = "";
                                for (i = r; parcali[i] == '\n' ||
parcali[i] == '\r'; i++) r = i;
                                for (i = r + 1; parcali[i] != '\n'; i++)
                                {

                                    sayi += parcali[i];
                                }
                                r = i;
                                richTextBox3.Text = sayi;
                                sayi = "";
                                for (i = r; parcali[i] == '\n' ||
parcali[i] == '\r'; i++) r = i;
                                for (i = r + 1; parcali[i] != '\n'; i++)
                                {

                                    if (parcali[i] == ' ') es++;
                                    sayi += parcali[i];

                                }
                                textBox1.Text = es.ToString();
                                richTextBox2.Text = sayi;

                            }
                            finally
                            {
                                if (file != null)
                                    file.Close();
                            }
                            Application.DoEvents();
                            for (int c = 0; c < 10; c++)
                            {
                                this.BackColor =
System.Drawing.Color.LemonChiffon;
                                BaslatButton.PerformClick();
                                this.BackColor = Form.DefaultBackColor;
                                Application.DoEvents(); }
                        }
                    }
                }
            }
        }
        private void Form1_FormClosed(object sender, FormClosedEventArgs e)
        {
            System.Diagnostics.Process.Start("sonuc.txt"); }}
```

**Ek 2 ABC sınıfının fonksiyon deklerasyonları "ABC.cs"**

```
class ABC : Form1
{
        public void ABCİterasyon(byte[,] dizi, double[] fitniss, double[]
        degerler, double[] agirlik, int[] fsay, int esya, int pop, int Flim,
        double limit)

        private double AbcFlimC(double GD,int Csize,int esya)

        public double[] AbcParam(double[] agirlik, double[] degerler, int esya)

        public bool relabc(double PFE, double NUF, int esya)

        private double GD(byte[,] dizi, int esya, int pop)

        private void gozcuAri(byte[,] Ydizi, double[] degerler, double[] agirlik,
        double[] fitness, int[] fsay, int esya, int pop, double limit)

        private void kasifAri(byte[,] dizi, double[] degerler, double[] agirlik,
        double[] fit, int[] fsay, int esya, int pop, double limit)

        private void fkontrol(byte[,] dizi, double[] degerler, double[] agirlik,
        double[] fit, int[] fsay, int esya, int pop, double limit, int fL)

        public void PopOlAbc(byte[,] gen, double[] agirlik, int[] fsay, int esya,
        int pop, double limit)

        public void PopOlAbcFromGa(byte[,] gen, double[] agirlik, int[] fsay, int
        esya, int pop, double limit)

        private void KomsuBul(byte[] komsu, byte[] dizi, double[] agirlik, int
        esya, double limit, int l)

        public void PopOlAbcFromSa(byte[] genSa,byte[,] gen, double[] agirlik,
        int[] fsay, int esya, int pop, double limit)
}
```

**Ek 3 GA sınıfının fonksiyon deklerasyonları "GA.cs"**

```
class GA : Form1
{
        public void GenetikİterasyonGenetikİterasyon(byte[,] dizi, double[] fitniss, double[]
        degerler, double[] agirlik, int esya, int pop, double mo, double limit)
        private double GAMoD(double GD, double PFE, int esya)
        private double GAND(double GD, double PFE, int esya)
        private bool relga(double GD, double NUF, int esya)
        public int gatoalg(double GD, int esya)
        public void PopDeGa(byte[,] gen,double[] fit ,double[] agirlik, int esya,
        int pop,int ypop, double limit)
        public void PopOlGa(byte[,] gen, double[] agirlik, int esya, int pop,
        double limit)
        public void PopOlGAFromABC(byte[,] gen, double[] agirlik, int esya, int
        pop, double limit)
        private void KomsuBul(byte[] komsu, byte[] dizi, double[] agirlik, int
        esya, double limit, int l)
        public void PopOlGaFromSa(byte[] genSa, byte[,] gen, double[] agirlik,
        int esya, int pop, double limit)
        public double[] GaParam(double[] agirlik, double[] degerler, int esya)
        private double GD(byte[,] dizi, int esya, int pop)
        private void Bestn(double[] fit, byte[,] dizi,int pop,int esya)
        private void caprazlama(double[] fit, byte[,] dizi, int esya, int pop)
        private void mutasyon(byte[,] dizi, double mo, int esya, int pop)
        private void tamir(byte[,] dizi, double[] degerler, double[] agirlik, int
        esya, int pop, double limit)
}
```

**Ek 4 SA sınıfının fonksiyon deklerasyonları "SA.cs"**

```
class SA: Form1

{

        public void Sa(byte[] gen, double[] agirlik, double[] degerler, double t,
        double sf, int esya, double limit)

        public double[] SaParam(double[] agirlik, double[] degerler, int esya)

        public double AmacFonk(byte[] dizi, double[] degerler, int esya)

        public void BasOlustur(byte[] gen, double[] agirlik, int esya, double
        limit)

        private void Komsuluk(byte[] dizi, double[] agirlik, int esya, double
        limit)

        public bool relas(double NCT, double NUF, int esya)

        public int satoalg(double[] agirlik, double[] degerler, int esya)

}
```

**Ek 5 Rasgele sayı üretim sınıfı "CRandom.cs"**

```
class CRandom
{
        private static RandomNumberGenerator r;
        public CRandom()
        {
            r = RandomNumberGenerator.Create();
        }
        public void GetBytes(byte[] buffer)
        {
            r.GetBytes(buffer);
        }
        public double NextDouble()
        {
            byte[] b = new byte[4];
            r.GetBytes(b);
            return (double)BitConverter.ToUInt32(b, 0) / UInt32.MaxValue;
        }
        public int Next(int minValue, int maxValue)
        {
            byte[] b = new byte[4];
            r.GetBytes(b);
            uint a = BitConverter.ToUInt32(b, 0);
            int son = (int)(a % (maxValue - minValue) + minValue);
            return son;
        }
        public int Next()
        {
            return Next(0, Int32.MaxValue);
        }
        public int Next(int maxValue)
        {
            return Next(0, maxValue);
        }

 }
```

**Ek 6 Bulanık küme sınıfı "Fuzzy.cs"**

```
class Fuzzy
    {
        public struct fset
        {

            public double a;
            public double b;
            public double c;
             public fset(double a,double b,double c)
            {
                 this.a=a;
                 this.b=b;
                 this.c=c;
            }
        }
        static public fset cor_u = new fset(0, 0, 0.33);
        static public fset cor_w = new fset(0.25, 0.5, 0.75);
        static public fset cor_c = new fset(0.67, 1, 1);

        static public fset p_size_s = new fset(0, 0, 400);
        static public fset p_size_l = new fset(300, 1000, Int64.MaxValue);

        static public fset c_rate_l = new fset(0.9,0.9,0.933);
        static public fset c_rate_m = new fset(0.9225,0.945,0.9675);
        static public fset c_rate_h = new fset(0.957,0.99,0.99);

        static public fset i_temp_l = new fset(0, 0, 75);
        static public fset i_temp_m = new fset(50, 100, 150);
        static public fset i_temp_h = new fset(125, 1000, 1000);

        static public fset NCT_vl = new fset(0, 0, 0.25);
        static public fset NCT_l = new fset(0,0.25,0.5);
        static public fset NCT_m = new fset(0.25,0.5 , 0.75);
        static public fset NCT_h = new fset(0.5, 0.75, 1);
        static public fset NCT_vh = new fset(0.75, 1, 1);

        static public fset NUF_vl = new fset(0, 0, 0.25);
        static public fset NUF_l = new fset(0, 0.25, 0.5);
        static public fset NUF_m = new fset(0.25, 0.5, 0.75);
        static public fset NUF_h = new fset(0.5, 0.75, 1);
        static public fset NUF_vh = new fset(0.75, 1, 1);

        static public fset Mo_l = new fset(0,0,0.01);
        static public fset Mo_h = new fset(0.05,0.15,0.20);
        static public fset Pop_l= new fset(100,100,175);
        static public fset Pop_m= new fset(150,200,250);
        static public fset Pop_h= new fset(225,300,300);

        static public fset GD_vl = new fset(0, 0, 0.25);
        static public fset GD_l = new fset(0, 0.25, 0.5);
        static public fset GD_m = new fset(0.25, 0.5, 0.75);
        static public fset GD_h = new fset(0.5, 0.75, 1);
        static public fset GD_vh = new fset(0.75, 1, 1);

        static public fset CS_l = new fset(50, 50, 90);
        static public fset CS_m = new fset(70, 100, 130);
        static public fset CS_h = new fset(110, 150, 150);

        static public fset L_l = new fset(0, 200, 350);
        static public fset L_m = new fset(250, 450, 650);
        static public fset L_h = new fset(550, 700, 900);
```

**Ek 5 Bulanık küme sınıfı "Fuzzy.cs" (devam)**

```
        static public fset PFE_vl = new fset(0, 0, 0.25);
        static public fset PFE_l = new fset(0, 0.25, 0.5);
        static public fset PFE_m = new fset(0.25, 0.5, 0.75);
        static public fset PFE_h = new fset(0.5, 0.75, 1);
        static public fset PFE_vh = new fset(0.75, 1, 1);

        static public fset Dfl_nh = new fset(0.5, 0.5, 0.75);
        static public fset Dfl_nm = new fset(0.5, 0.75, 1);
        static public fset Dfl_nc = new fset(0.75, 1, 1.25);
        static public fset Dfl_pm = new fset(1, 1.25, 1.5);
        static public fset Dfl_ph = new fset(1.25, 1.5, 1.5);

        static public fset DMo_nh = new fset(0.5, 0.5, 0.75);
        static public fset DMo_nm = new fset(0.5, 0.75, 1);
        static public fset DMo_nc = new fset(0.75, 1, 1.25);
        static public fset DMo_pm = new fset(1, 1.25, 1.5);
        static public fset DMo_ph = new fset(1.25, 1.5, 1.5);

        static public fset DN_nh = new fset(0.5, 0.5, 0.75);
        static public fset DN_nm = new fset(0.5, 0.75, 1);
        static public fset DN_nc = new fset(0.75, 1, 1.25);
        static public fset DN_pm = new fset(1, 1.25, 1.5);
        static public fset DN_ph = new fset(1.25, 1.5, 1.5);

        static public fset NTUF_vl = new fset(0, 0, 0.25);
        static public fset NTUF_l = new fset(0, 0.25, 0.5);
        static public fset NTUF_m = new fset(0.25, 0.5, 0.75);
        static public fset NTUF_h = new fset(0.5, 0.75, 1);
        static public fset NTUF_vh = new fset(0.75, 1, 1);

        public double mom(double[] a)
        {
            return (a[0] + a[1]) / 2;
        }
        public double[] itrimf(double y, fset f)
        {
            double[] son = new double[2];
            son[0] = f.a + (y * (f.b - f.a));
            son[1] = f.c - (y * (f.c - f.b));
            return son;
        }
        public double trimf(double x, fset f)
        {

            return max(min(((x - f.a) ==(f.b - f.a)) ? 1 : (x - f.a) / (f.b -
f.a),((f.c - x) == (f.c - f.b)) ? 1 : (f.c - x) / (f.c - f.b)), 0);
        }
        public double min(double a, double b)
        {
            return a < b ? a : b;
        }
        public double max(double a, double b)
        {
            return a > b ? a : b;
        }

    }
```