T.C.
SAKARYA ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

KLASİK VE ZEKİ OPTİMİZASYON TEKNİKLERİ İLE PROBLEMLERİN ÇÖZÜMLENMESİ VE KARŞILAŞTIRILMASI

YÜKSEK LİSANS TEZİ

End.Müh. Ünal Atakan KAHRAMAN

Enstitü Anabilim Dalı : ENDÜSTRİ MÜHENDİSLİĞİ

Tez Danışmanı : Prof. Dr. Harun TAŞKIN

HAZİRAN 2010

T.C.
SAKARYA ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

KLASİK VE ZEKİ OPTİMİZASYON TEKNİKLERİ İLE PROBLEMLERİN ÇÖZÜMLENMESİ VE KARŞILAŞTIRILMASI

YÜKSEK LİSANS TEZİ

End.Müh. Ünal Atakan KAHRAMAN

Enstitü Anabilim Dalı : ENDÜSTRİ MÜHENDİSLİĞİ

Bu tez 03 / 06 /2010 tarihinde aşağıdaki jüri tarafından Oybirliği ile kabul edilmiştir.

Prof.Dr. Harun TAŞKIN	Doç.Dr. Cemalettin KUBAT	Yrd.Doç.Dr. Yılmaz UYAROĞLU
.....	.....	.....
Jüri Başkanı	Üye	Üye

TEŞEKKÜR

Çalışmayı hazırlarken maddi manevi her türlü destekleriyle daima yanımda olan aileme, lisans öğrenimim ve yüksek lisans tez danışmanlığımın başlangıcından beri, bilgi ve tecrübelerini paylaşarak bana yol gösteren değerli Danışman Hocam Prof. Dr. Harun TAŞKIN' a, tavsiyeleriyle çalışma azim ve kararlılığımı daima arttıran değerli Hocalarım Prof.Dr. Mehmet DUMAN, Yrd.Doç.Dr. Esat PINARBAŞI, ve Prof.Dr. Mustafa Fehmi TÜRKER' e teşekkürü borç bilirim.

İÇİNDEKİLER

BÖLÜM 2.

KLASİK OPTİMİZASYON TEKNİKLERİNİN SINIFLANDIRILMASI, PROBLEM TİPLERİ VE ÇÖZÜMLERİ

BÖLÜM 3.

ZEKİ OPTİMİZASYON TEKNİKLERİNİN İNCELENMESİ

KISALTMALAR LİSTESİ

AMS	: Arızalı Makine Sayısı
API	: Application Program Interface (Uygulama Programı Arayüzü)
Bkz.	: Bakınız
BLF	: Geriye Yayılım Ağırlık/Bias Öğrenme Fonksiyonu
BTF	: Varsayılan Geriye Yayılım Ağ Eğitim Fonksiyonu
BFGS	: Broyden-Fletcher-Goldfarb-Shanno Formülü
GA	: Genetik Algoritmalar
IDL	: Interface Definition Language (Arayüz Tanımlama Dili)
IEEE	: Institute of Electrical and Electronics Engineers (Elektrik ve Elektronik Mühendisleri Enstitüsü)
KKO	: Karınaca Koloni Optimizasyonu
LVQ	: Lineer Vector Quantization (Lineer Vektör Nicelemesi)
MSE	: Mean Squared Error (Ortalama Kareli Hata)
MLP	: Multi Layer Perceptions (Çok Katmanlı)
OMDT	: Object-Model Development Tool (Nesne Modeli Geliştirme Aracı)
Ör.	: Örneğin
TSK	: Takagi – Sugeno – Kang Bulanık Sistemler
PF	: Varsayılan Performans Fonksiyonu
SEO	: Search Engine Optimization (Arama Motoru Optimizasyonu)
SO	: Sürü Optimizasyonu
SP	: Stokastik Programlama
SR1	: Symmetric Rank 1 Formülü (Simetrik Mertebe 1 Formülü)
TA	: Tabu Arama
TBA	: Tavlama Benzetimi Algoritması
vb.	: Ve Benzeri
vs.	: Vesaire

XML : Extensible Markup Language (Genişletilebilir Biçimleme Dili)

YSA : Yapay Sinir Ağları

YMS : Yedek Makine Sayısı

ŞEKİLLER LİSTESİ

TABLOLAR LİSTESİ

ÖZET

Anahtar Kelimeler: Klasik Optimizasyon, Zeki Optimizasyon, Karşılaştırma, Optimal Çözüm

Bir işin en iyi yolun seçilerek başarılması fikri, uygarlık tarihi kadar eskidir. Bu başarımı, klasik ve zeki optimizasyon teknikleri üzerinden incelemek ve bu noktada karşılaştırma bakış açısının kazandırılması tez çalışmasının temelini teşkil etmektedir.

Optimizasyon teknikleri açısından geçilen zaman periyotu süresince sezgisel araçlar geliştirilmiş, bu araçlar çözümü zor veya olanaksız olan optimizasyon problemlerini çözülebilir hale getirmiştir. Bu araçlar birbirleriyle kaynaşırken, bilgi elemanları ve de istatistiksel analiz gibi daha geleneksel yaklaşımlarla bütünleşerek çok daha zorlayıcı problemleri çözme hedefi edinilmiştir. Zeki optimizasyon tekniği kapsamında bahsedilen araçlar ile çözümler geliştirmek iki önemli avantaj sağlar.

- Geliştirme süresi klasik yaklaşımlara göre çok daha kısadır.
- Kayıp verilere göreceli olarak duyarsız olması sonucu klasik yaklaşımlara göre daha güçlüdür.

Buradan hareketle klasik optimizasyon yaklaşımları genel bir bakış ile örneklerle değerlendirilmiş, zeki optimizasyon tekniklerindeki temel bilgiler, hesap ve uygulamanın nasıl yapıldığı algoritmalarla gösterilmeye çalışılmıştır. Amaç yukarıda belirtilen avantajları problem örnekleriyle gösterebilmektir.

THE ANALYSIS AND COMPARISON OF PROBLEMS BY CLASSICAL AND INTELLIGENT OPTIMIZATION TECHNIQUES

SUMMARY

Keywords: Classical Optimization, Intelligent Optimization, Comparison, Optimal Solution

The idea of achieving an issue by choosing the best solution way is as olden as civilization history. Analysing that kind of achievement through classical and intelligent optimization technics and comparing these technics come to the heart of this study.

In the last of optimization technics some heuristic tools have been developed. These tools make the optimization problems, of which solution is difficult or impossible, easy to solve. While these tools merge with each other, they unite conventional approaches such as information elements and statistical analysis. The purpose of these merges and unifications is to solve much more challenging problems. There are two important advantages of developing solutions by the help of these tools, which are mentioned in the context of intelligent optimization technics.

- The time for developing these solutions is much shorter in comparison with classical approaches.

- This approach is stronger than classical approaches as it is relatively insensitive to lost data.

In this context classical optimization approaches are reviewed with examples by a general view and it is purposed to denote the fundamental information about intelligent optimization technics and demonstrate how to compute and apply by using algorithms. The object is to show the advantages indicated above by the help of problem examples.

BÖLÜM 1. GİRİŞ

1.1. Giriş

Optimizasyon, verilen şartlar altında en iyi sonucun elde edilmesi işlemidir. Bir fonksiyonun maksimum yada minimum değerini, verilen şartları da sağlayarak araştırmaya optimizasyon problemi adı verilir. Optimizasyon problemlerini incelemek ve çözmek için geliştirilen yöntemlerin tümüne optimizasyon teknikleri adı verilmektedir.

1.2. Tanım

Matematik problemi, belirli kısıtlar altında bir amaç fonksiyonunun optimize edilmesinden oluşmaktadır. Diğer bir deyişle, karar değişkenleri olarak nitelendirilen fonksiyon değişkenlerinin kısıtların tümünü sağlayan (uygun çözüm bölgesinde bulunan) ve amaç fonksiyonunu optimize eden sayısal değerlerini bulma problemidir. Tipik bir matematik program aşağıdaki gibi ifade edilebilir; n tane değişken sayısı ve m tane kısıt olmak üzere; [1]

Amaç Fonksiyonu $Z = f(x_1, x_2, \dots, x_n)$

$$\text{Kısıtlar} \begin{matrix} g_1(x_1, x_2, \dots, x_n) \\ g_2(x_1, x_2, \dots, x_n) \\ \dots\ \dots\ \dots\ \dots\ \dots\ \dots\ \dots \\ g_m(x_1, x_2, \dots, x_n) \end{matrix} \Bigg\} <, \leq, =, >, \geq \begin{cases} b_1 \\ b_2 \\ \dots \\ b_m \end{cases} \quad [2]$$

Bu ifadede, m sayıda farklı kısıt <, ≤ , = , >, ≥ sembollerinden birisini içerebilir. Her g_i fonksiyonu ve g_i katsayıları sıfır seçilirse kısıtsız matematik programlar elde edilir.

Kâr, getiri, fayda ve benzeri gibi kavramlar amaç fonksiyonunda yer aldığında amaç fonksiyonunun maksimizasyonu; maliyet, gider ve benzeri gibi kavramlar yer

aldığında ise amaç fonksiyonunun minimizasyonu yapılır. g_i fonksiyonlarının her biri birer kısıt belirtmektedir. Kısıt sayısında herhangi bir sınır bulunmamaktadır. Kısıtların hepsi birlikte bir uygun çözüm bölgesi belirlerler [1].

Optimal çözüm değeri veya değerleri bu bölgeye ait bir değer olmaktadır. Kısıtlar sınırlayıcı şartların ifadeleridir. İşletme ve ekonomi problemlerinde sınırlayıcı şartların varlığını görebilmek oldukça kolaydır. Örneğin, üretilmesi planlanan ürünler için hammadde, işçilik, makine zamanı, stoklama alanı gibi sınırlamalar kısıtlar olarak ifade edilirler. Bazı özel problemlerde amaç fonksiyonu olmayabilmektedir. Optimizasyonun söz konusu olmadığı böylesi modellerde sadece uygun bir çözümün varlığı yeterli olmaktadır.

1.3. Optimizasyonun Tarihi

Bir işin en iyi yolun seçilerek başarılması fikri uygarlık tarihi kadar eskidir. Örneğin, Yunan tarihçisi Herodotus' a göre, Mısırlılar Nil nehrinin her yıl taşması sonucu arazi sınırlarının yeniden belirlenmesi ve yeni sınırlara göre vergilendirme işleminin en iyi yolla yapılabilmesi için çaba sarf etmişlerdir. Bu çabalar, ölçme ve karar verme aracı olarak düzlem geometrisinin temel kavramlarının oluşturulmasına yol açmıştır. Mısırlılar, Nil nehrinin bahar dönemlerindeki yıllık taşmalarında nehir kıyısından toplu halde uzaklaşıp sular çekildiğinde yine büyük topluluklar halinde geri dönüyorlardı. Çekilme işlemi çok kısa sürede yapılamamaktaydı. Bunun için günlerce önceden halk uyarılmalıydı. Bu amaçla, Mısırlılar en iyi çekilme zamanını hesaplayabilmek için bir tür takvim bile geliştirmişlerdi. Söz konusu takvimi de sayma ve geometri konusundaki birikimlerini kullanarak yapmışlardı. Optimizasyon günümüzde üretim ve hizmet sektöründe yöneylem araştırması, yapay zeka, finansal alt başlıklarında kullanılan önemli bir araçtır.

1.4. Uygulama Alanlarına Göre Çeşitli Optimizasyon Kavramları

Optimizasyona konu olan bazı problemleri alanlarına göre şöyle örneklendirilebilir.

Maliyet optimizasyonu: Kârın maksimizasyonu veya maliyetin minimizasyonu; optimizasyon bir işletmede kazancı maksimize etmek ya da maliyeti minimize etmek için kullanılabilir. Örneğin, mevcut kaynaklarla daha fazla üretim gerçekleştirerek, birim başına maliyeti düşürebilir, kısıtları ihlal etmeksizin belirli özellikteki ürünü daha ucuz hammadde ile üretilebilir.

Proses optimizasyonu (process optimization): Bazı kısıtları ihlal etmeksizin belirli parametreler kümesini optimize edecek şekilde bir prosesi ayarlama sürecidir. Bu bağlamda en temel hedefler maliyeti minimize etme, çıktı maksimizasyonu, kar maksimizasyonu vb. Burada optimal performansı etkileyecek üç parametre vardır.

a. **İşlem optimizasyonu:** İşlem yöntemleri kişiden kişiye vardiyadan vardiyaya değişkenlik gösterebilir. Tezgahların otomasyonu değişkenliği azaltmada önemli ölçüde yardımcı olabilir. Fakat kontrol işlemleri operatörlerce yapılıp, tezgahlar el yordamıyla çalıştırılıyorsa otomasyonun gücü azalır. İşlem optimizasyonu değişkenliğin azaltılmasını amaç edinerek bunların en verimli bir şekilde kullanılmasını sağlar.

b. **Kontrol optimizasyonu:** Kimyasal ya da petrol rafinerisi gibi üretim tezgahlarında yüzlerce kontrol döngüsü olabilir. Her bir kontrol döngüsü belli bir sıcaklığı sürdürme gibi prosesin bir parçasının kontrolü için gerçekleştirilir. Eğer kontrol döngüsü doğru bir biçimde dizayn edilmeyip, ayarlanmazsa proses optimum düzeyinin altında çalışır.

c. **Ürün optimizasyonu (product optimization):** Belirli bir ürünü talep edilir hale getirmek için kalitesini iyileştirme bağlamında yapılacak değişikliklerin yöntemleridir. Bir ürün birçok özelliğe sahiptir. Örneğin bir soda şişesi çeşitli paketleme opsiyonlarına, meyve aromalarına, besin değerlerine sahiptir. Küçük değişiklikler yaparak bir ürünü optimize etmek mümkündür. Amaç, ürünü daha

istenir hale getirmek ve “ satın alma isteği (purchase intent) ”, “ inandırıcılık (believability)”, “ satın alma sıklığı (frequency of purchase) ” gibi ölçümleri arttırmaktır.

Çok değişkenli ürün optimizasyonu: En yaygın kullanılan yöntemlerden birisidir. Bu yöntemde çoklu ürün özellikleri tanımlanır ve müşterilerle test edilir. Farklı özellikler arasındaki etkileşim (örneğin müşteriler belli aromaları belli paket renkleriyle ilişkilendirebilir) nedeniyle son zamanlarda ürün optimizasyonu için Veri Zarflama Analizi (Data envelope analysis) gibi zeki optimizasyon teknikleri (Evolutionary Optimization Techniques) kullanılmaya başladı.

Bilgisayar optimizasyonu: Bilgisayar bilimlerinde optimizasyon, sistemin daha az kaynak gerektirerek daha etkin çalışmasını sağlayacak biçimde modifiye edilmesi prosesidir. İşlem zamanının azaltılması, bant genişliğinin azaltılması, hafıza gereksiniminin en aza indirilmesi, compile edilen program kodunun etkinliğinin artırılmasıdır. Örneğin, bir bilgisayar programı daha hızlı biçimde (run) yada daha az hafıza gerektirecek biçimde optimize edilebilir.

Arama motorları optimizasyonu: Arama motorlarından bir web sitesine yönlenen trafiğin hacim ve kalitesinin algoritmalar yoluyla en iyileştirilmesi prosesidir. Tipik olarak, bir sitenin sayfasının arama listesindeki sırasının en başa getirilmesi arzulanır. Internet pazarlama stratejisinde, arama motoru optimizasyonu (SEO) arama motorunun nasıl çalıştığını ve insanların ne için arama yaptığını dikkate alır. Bir web sitesini optimize etme, temel olarak web sitesinin içeriğini yazma ve HTML kodunun her ikisinin anahtar kelimelerle ilgisini artırma ve arama motorunun indeksleme faaliyetinin önündeki bariyerleri kaldırma hedefine yöneliktir.

1.5. Optimizasyonun Bileşenleri

Bir optimizasyon probleminin 3 temel bileşeni vardır.

1.5.1. Amaç fonksiyonu

Maksimum ya da minimum yapılmak istenen fonksiyon olarak tanımlanır. Örneğin, bir imalat ya da üretim işleminde kâr maksimum yada maliyet minimum yapılmak istenebilir. Deneysel verilerin bilinen bir modele uydurulması probleminde gözlenen veri ile tahmin edilen değer arasındaki sapma minimum yapılmak istenebilir. Bu durumda modelin bilinmeyen parametreleri tahmin edilir. Ya da bir oto ön kapı aynasının tasarlanması probleminde aynanın dayanıklılığı maksimum yapılmak istenebilir [3].

1.5.2. Bilinmeyenler ya da değişkenler

Amaç fonksiyonun değerini etkileyen değişkenlerdir. Üretim probleminde kullanılan kaynak miktarı ya da işlemler için kullanılan zaman değişken olarak alınabilir. Verilerin verilen modele uydurulması probleminde değişkenler modelin parametreleri olur. Oto aynasını tasarı probleminde oto kapı aynasının boyutu, şekli değişkenlerdir [3].

1.5.3. Kısıtlayıcılar

Bilinmeyenlerin yada değişkenlerin, belirli değerleri almasına ve belirli değerleri de almamasına yarayan faktörlerdir. Bu bileşenler her zaman gerekli mi şeklinde akıllara bir soru gelebilir. Bunun yanıtı şu şekilde verilebilir. Hemen hemen her optimizasyon probleminin bir amaç fonksiyonu vardır. Hatta bazen birden fazla amaç fonksiyonunun olduğu optimizasyon problemleri de bulunmaktadır. Örneğin oto aynası tasarı probleminde oto kapı aynasının hem dayanıklılığı maksimum, hem de aynanın ağırlığı minimum yapılabilir. Herhangi bir kriter fonksiyonunu göz önüne almadan yalnızca verilen şartları sağlayan optimizasyon problemleri de vardır.

Değişkenler; bunlar optimizasyon problemlerinin olmazsa olmazlarıdır, yani temel bileşenleridir, bunlar olmaksızın amaç fonksiyon ve kısıtlayıcılar oluşturulamaz.
Kısıtlayıcılar; Hem kısıtlayıcıların olduğu hem de olmadığı optimizasyon problemleri vardır. Bunlarla ilgili kısıtsız optimizasyon ve kısıtlı optimizasyon alanları bulunmaktadır.

1.6. Tezin İçeriği

Bu tez çalışması 4 ana bölümden oluşmaktadır. Birinci bölümde optimizasyonun tanımı, optimizasyonun tarihi, çeşitli optimizasyon terimleri ve optimizasyonun kavramları anlatılmıştır. İkinci bölümde klasik optimizasyon tekniklerinin sınıflandırılması ve problem çözümleriyle açıklanması yapılmıştır. Üçüncü bölümde ise zeki optimizasyon teknikleri belirtilerek, problem çözümlerinin bu teknikler ile yapılması yer almıştır. Dördüncü bölüm sonuçlar ve öneriler bölümüdür. Burada zeki optimizasyonun ikinci ve üçüncü bölümde anlatılanların ışığında, klasik optimizasyona göre mukayesesi yapılarak sonuçlar belirtilmiş ve önerilerde bulunulmuştur.

1.7. Tezin Amacı

Bu tez çalışmasının amacı optimizasyon tekniklerini, klasik ve zeki teknikler bazında problem incelemeleri yaparak çözümlerini gerçekleştirmek ve zeki optimizasyonun çıkış noktasını yakalayarak, kullanılan zeki araçların klasik tekniklere göre üstün ve zayıf yönlerini görmeyi sağlayacak bakış açısını kazandırmaktır.

BÖLÜM 2. KLASİK OPTİMİZASYON TEKNİKLERİNİN SINIFLANIDIRILMASI, PROBLEM TİPLERİ VE ÇÖZÜMLERİ

Verilen sürekli ve türevlenebilir bir fonksiyonun diferansiyel matematik kuralları göz önüne alınarak maksimum yada minimum değerinin araştırılması problemine klasik optimizasyon problemi adı verilir. Bu çerçevede klasik optimizasyon teknikleri kısıtlı ve kısıtsız optimizasyon olarak ikiye ayrılır. Sınıflandırma aşağıdaki kapsamda incelenecektir.

2.1. Klasik Optimizasyon

2.1.1. Kısıtlı optimizasyon teknikleri

Doğrusal optimizasyon

Şebeke optimizasyonu

Tamsayılı optimizasyon

Dinamik optimizasyon

Stokastik optimizasyon

Kuadratik (Karesel) optimizasyon

2.1.2. Kısıtsız optimizasyon teknikleri

Golden section (Altın Oran) metodu

Newton metodu

Sabit kesen (Regula Falsi) yöntemi

Gradient arama

Lagrange çarpanı

Newton benzeri algoritması (Quasi Newton)

2.1.1. Kısıtlı optimizasyon teknikleri

- **Doğrusal optimizasyon**

Doğrusal eşitsizlikler sistemi şeklindeki bir problemin incelenmesi Fourier' in çalışmalarına kadar dayanmaktadır. 1920'lerde Sovyet Rusya'da tüm ekonomi planlaması konuları pratikte ön plana geçmişken teorik olarak tüm ekonominin nasıl planlanabileceğini göstermek için yapılan teorik çalışmalar arasında Leonid Kantoriviçin katkısı ilk defa bir doğrusal optimizasyon probleminin açıkça ortaya çıkarılmasına yol açmıştır. Daha sonra ikinci dünya savaşı sırasında grup başkanı George Dantzig olan, ABD' de ortaya çıkan lojistik tahsis sorunlarını incelemek için kurulan bir araştırma grubu, bu türlü sorunların çözülmesi için doğrusal optimizasyon probleminin tanımlanması gereğini ortaya çıkartmışlar ve bu türlü problemlerin çözümü için simpleks algoritması adını verdikleri bir çözüm sistemi ortaya atmışlardır. 1947'de John Von Neumann, özellikle oyunlar teorisi ile de ilgileniyorken, dualite teorisini geliştirmiştir. 1947'den sonra özellikle geliştirilen bilgisayar uygulamaları ile birlikte özellikle büyük özel sanayi birimleri ve büyük devlet projeleri için birçok doğrusal programlama problem tanımlanmış ve simpleks algoritması ile çözülüp pratikte kullanılmaya başlanmıştır. Örneğin petrol rafine şirketlerinin günlük üretim planlamaları ve çok girdili ve çok çıktılı üretim karışımı planlamaları için doğrusal programlama çözümlerini devamlı olarak kullanmaya başlamışlardır. Bu zamana kadar doğrusal programlamaya yaptıkları katkılar nedeni ile Kantroviç, Dantzing ve Von Neumann'a 1975'de Nobel Ekonomi ödülü verilmiştir. Bu alanda çok daha önemli teorik ve pratik gelişmeler 1984'te Narendra Karmarkar'ın doğrusal programlama problemlerin çözülmesi için (simpleks algoritması yerine) içsel nokta yöntemi ortaya atması ile başlamıştır. Problemin bilgisayarla simpleks algoritması kullanılarak çözülmesi saniyeler bile almaz. Doğrusal programlama kuramı arkasında bulunan teori, kontrol edilmesi gereken mümkün en iyi çözüm sayısını çok etkili şekilde azaltmaktadır [1-3].

Sınırlı kaynakların kullanımını optimum kılmak için tasarlanmış bir matematiksel modelleme yöntemidir [5]. Temel elemanları;

Belirlenecek Karar Değişkenleri
Optimum Kılınacak Amaç (Hedef) Fonksiyonu
İçinde Bulunulan Kısıtlardır

Doğrusal Optimizasyon modellerindeki bazı varsayımları ise söyle sıralayabiliriz.

Doğrusallık: Bu varsayım sistem içerisindeki girdi ve çıktılar arasında doğrusal bir ilişkinin bulunduğunu gösterir.

Toplanabilirlik: Bu varsayım değişik üretim faaliyetlerine kaynak olan üretim girdilerinin toplamının her bir işlem için ayrı ayrı kullanılan girdilerinin toplamına eşit olduğunu gösterir.

Sınırlılık Varsayımı: Kullanılan kaynaklar belli değerlerde sınırlandırılmıştır.

Negatif Olmama: Değişkenlerin değeri sıfır veya pozitif bir değer almalıdır.

Doğrusal optimizasyon yönteminin çözüm metodları aşağıdaki ana eksendedir.
Amaç fonksiyonu maksimizasyon veya minimizasyonu hedefler.

Grafik Çözüm Metodu
Simpleks Metod
  M Tekniği
  İki Aşamalı Yöntem
Dual Simpleks Metod

Doğrusal optimizasyoda bilinmeyen sayısı 3'ü aştığında geometrik yöntemle çözüme ulaşılamaz. Bu durumda daha farklı çözüm yöntemleri kullanılmalıdır. 1940' lardan bu yana bilgisayar teknolojisindeki hızlı değişiklikler, çok bilinmeyenli denklemlerin çözümünü olanaklı kılmıştır.

Uygulamada karşılaşılan modeller, yüzlerce hatta binlerce kısıt içerebilir, böyle modelleri çözmenin en mantıklı yolu, çözüm metodlarını uygun bir bilgisayar yazılımına çevirerek kullanmaktır. Bu noktadan hareketle bir maksimizasyon problemi ve problemlerinin çözümü aşağıda verilen örnekle gösterilecektir.

Örnek: Bir firma hammadde ve işçilik kullanarak dört ayrı ürün üretmektedir. Bu ürünlere ait gerekli bilgiler tablo 2.1. de ki gibidir [4].

Tablo 2.1. Ürünlere Ait Hammadde, İşçilik, Satış Fiyatı Tablosu

	Ürün 1	**Ürün 2**	**Ürün 3**	**Ürün 4**
Hammadde (br)	2	3	4	7
İşçilik (saat)	3	4	5	6
Satış fiyatı (TL)	4	6	7	8

Halihazırda 4600 br hammadde ve 5000 işçilik saati mevcuttur. Ürünlere olan toplam müşteri talebi 950 adettir. Müşteriler ürün 4 ten en az 400 adet olmasını talep etmektedir. Firmanın kârını en çok yapmak için gerekli doğrusal optimizasyon nasıl olmalıdır.

Çözüm

x_1; Satıştaki ürün1 adedi

x_2; Satıştaki ürün2 adedi (Karar değişkenleri)

x_3; Satıştaki ürün3 adedi

x_4; Satıştaki ürün4 adedi olmak üzere

Zmax $= 4x_1 + 6x_2 + 7x_3 + 8x_4$ (Amaç Fonksiyonu)

$x_4 \geq 400$

$2x_1 + 3x_2 + 4x_3 + 7x_4 \leq 4600$

$3x_1 + 4x_2 + 5x_3 + 6x_4 \leq 5000$

$x_1 + x_2 + x_3 + x_4 = 950$ (Kısıtlar)

$x_1 \geq 0$

$x_2 \geq 0$

$x_3 \geq 0, x_4 \geq 0$

TORA yazılımının doğrusal optimizasyon alt modülü olan Lindo da problem çözülecektir.

Verilerin aktarılması

Max 4x1+6x2+7x3+8x4

st

2x1+3x2+4x3+7x4<=4600

3x1+4x2+5x3+6x4<=5000

x1+x2+x3+x4=950

x4>=400

x1>=0

x2>=0

x3>=0

End

Çıktıların Analizi

Optımum Found At Step 4 (4 adımda optimum çözüm bulundu)

Objectıve Functıon Value (amaç fonksiyonun değeri)

Zmax=6650.000 TL Optimum Çözüm

Çözüm, temel olarak çıktı ekranındaki dört başlık altında incelenmiştir

A) Çözüme giren kaynaklar

Tablo 2.2. Çözüme Giren Kaynaklar (maksimum probleminde)

VARIABLE (değişken)	VALUE (değer)	REDUCED COST (Çözüme girmeyen kısıt, üretiminde karlılık olmayan kaynak)
X1	0.000000	1.000000
X2	400.000000	0.000000
X3	150.000000	0.000000
X4	400.000000	0.000000

B) Üstence (Artıklar) ve gölge fiyatlar

Tablo 2.3. Artıklar ve Gölge Fiyatlar (maksimum probleminde)

ROW (satır)	SLACK OR SURPLUS (artıklar veya âtıl kapasite)	DUAL PRICES (gölge fiyatlar)
2	0.000000	1.000000
3	250.000000	0.000000
4	0.000000	3.000000
5	0.000000	-2.000000

Açıklama

2. kaynaktaki (hammadde) 1 br'lik artış optimum çözümde 1 br lik artmaya (kârlılık) sebep olur.

3. kaynağın (işçilik saati) atıl kalan miktarı ve kaynaktaki 1br'lik değişim optimum çözümü (kârlılığı) değiştirmediği

4. kaynaktaki (talep) 1 br'lik artış optimum çözümü (kârlılığı) 3 br arttırır.

5. kaynaktaki (ürün 4 adedi) 1 br'lik artış optimum çözümde (kârlılıkta) 2 br'lik azalmaya sebep olur.

C) Optimalliği bozmayacak amaç fonksiyonundaki katsayı aralıkları

Tablo 2.4. Katsayı Aralıkları (maksimum probleminde)

Değişkenler	COEF (Katsayılar)	INCREASE (Artış)	DECREASE (Azalış)	ARALIKLAR
X1	4.000000	1.000000	INFINITY	$-\infty \leq C_1 \leq 5$
X2	6.000000	0.666667	0.500000	$5{,}5 \leq C_2 \leq 6{,}666667$
X3	7.000000	1.000000	0.500000	$6{,}5 \leq C_3 \leq 8$
X4	8.000000	2.000000	INFINITY	$6 \leq C_4 \leq +\infty$

D) Optimalliği bozmayacak sağ taraf değişkenlerinin değişimi

Tablo 2.5. Sağ Taraf Değişkenlerinin Değişimi (maksimum probleminde)

RHS	INCREASE (Artış)	DECREASE (Azalış)	Aralıklar
4600.000000	250.000000	150.000000	$4450 \leq b_1 \leq 4850$
5000.000000	INFINITY	250.000000	$4750 \leq b_2 \leq +\infty$
950.000000	50.000000	100.000000	$850 \leq b_3 \leq 1000$
400.000000	37.5000000	125.000000	$275 \leq b_4 \leq 437{,}5$

Doğrusal optimizasyon minimizasyon problemi için çözüm aşağıdaki gibidir.

Örnek: Bir otomobil firması dört farklı fabrikada otomobil üretmektedir. Her bir fabrikada üretim maliyetleri tablodaki gibidir [3-4].

Tablo 2.6. Ürünler Ait Hammadde İşçilik Satış Fiyatı Tablosu (minimum probleminde)

Fabrika	**Maliyetler (TL/adet)**	**İşçilik (saat/adet)**	**Hammadde (br/adet)**
1	15	2	3
2	10	3	4
3	9	4	5
4	7	5	6

Üçüncü fabrikada en az 400 otomobil üretilmelidir. Dört fabrikada toplam 3300 saat işçilik ve toplam 4000 br hammadde mevcuttur. Firma 1000 aracı en az maliyetle hangi fabrikalarda üretmelidir.

Çözüm

x_1; 1. Fabrikada üretilen otomobil adedi

x_2; 2. Fabrikada üretilen otomobil adedi (Karar değişkenleri)

x_3; 3. Fabrikada üretilen otomobil adedi

x_4; 4. Fabrikada üretilen otomobil adedi olmak üzere

Zmin= $15x_1 + 10x_2 + 9x_3 + 7x_4$ (Amaç Fonksiyonu

$2x_1 + 3x_2 + 4x_3 + 5x_4 \leq 3300$

$3x_1 + 4x_2 + 5x_3 + 6x_4 \leq 4000$

$x_1 + x_2 + x_3 + x_4 = 1000$ (Kısıtlar)

$x_3 \geq 400$

$x_1 \geq 0$

$x_2 \geq 0$

$x_4 \geq 0$

TORA yazılımının doğrusal programlama alt modülünde problem çözülecektir.

Verilerin aktarılması

Min 15x1+10x2+9x3+7x4

st

2x1+3x2+4x3+5x4<=3300

3x1+4x2+5x3+6x4<=4000

x1+x2+x3+x4=1000

x3>=400

x1>=0

x2>=0

x4>=0

end

Çıktıların Analizi

Optımum Found At Step 4 (4 adımda optimum çözüm bulundu)

Objectıve Functıon Value (amaç fonksiyonun değeri)

Zmin = 11600.00 TL Optimum Çözüm

A) Çözüme giren kaynaklar

Tablo 2.7. Çözüme Giren Kaynaklar (minimum probleminde)

VARIABLE (değişken)	VALUE (değer)	REDUCED COST (Çözüme girmeyen kısıt, üretim maliyetini azaltmayan kaynak)
X1	400.000000	0.000000
X2	200.000000	0.000000
X3	400.000000	0.000000
X4	0.000000	7.000000

B) Üstence (Artıklar) ve gölge fiyatlar

Tablo 2.8. Artıklar ve Gölge Fiyatlar (minimum probleminde)

ROW (satır)	SLACK OR SURPLUS (artklar veya âtıl kapasite)	DUAL PRICES (gölge fiyatlar)
2	300.000000	0.000000
3	0.000000	5.000000
4	0.000000	-30.000000
5	0.000000	4.000000

Açıklama

2. kaynağın (işçilik) atıl kalan miktarı (300saat) ve kaynağın maliyet üzerindeki etkisizliği

3. kaynağın (hammadde) atıl kalan miktarı (0) ve kaynaktaki 1 br'lik artış maliyeti 5 TL azaltır.

4. kaynağın (otomobil adedi) atıl kalan miktarı (0) ve kaynaktaki 1br'lik artış maliyet üzerinde 30 TL artışa sebep olur.

5. kaynağın (3 nolu fabrikada üretilecek otomobil adedi) atıl kalan miktarı (0) ve kaynaktaki 1 br' lik artış maliyeti 4 TL azaltır.

C) Optimalliği bozmayacak amaç fonksiyonundaki katsayı aralıkları

Tablo 2.9. Katsayı Aralıkları (minimum probleminde)

Değişkenler	COEF (Katsayılar)	INCREASE (Artış)	DECREASE (Azalış)	Aralıklar
X1	15.000000	INFINITY	INFINITY	$11{,}5 \leq C_1 \leq +\infty$
X2	10.000000	2.000000	0.500000	$-\infty \leq C_2 \leq 12$
X3	9.000000	INFINITY	0.500000	$5 \leq C_3 \leq +\infty$
X4	7.000000	INFINITY	INFINITY	$0 \leq C_4 \leq +\infty$

D) Optimalliği bozmayacak sağ taraf değişkenlerinin değişimi

Tablo 2.10. Sağ Taraf Değişkenlerinin Değişimi (minimum probleminde)

RHS	INCREASE (Artış)	DECREASE (Azalış)	Aralıklar
3300.000000	INFINITY	300.000000	$3000 \leq b_1 \leq +\infty$
4000.000000	300.000000	200.000000	$3800 \leq b_2 \leq 4300$
1000.000000	66.666664	100.000000	$390 \leq b_3 \leq 1066$
400.000000	100.000000	400.000000	$0 \leq b_4 \leq 500$

Yukarıda gösterildiği gibi çözüm, temel olarak çıktı ekranındaki dört başlık altında incelenmiştir.

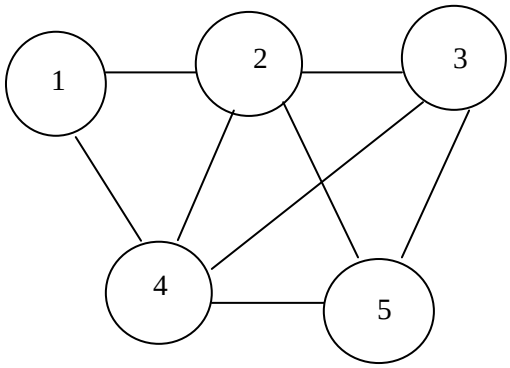
- **Şebeke optimizasyonu**

Şebeke olarak uygun bir biçimde modellenip çözülebilen çok sayıda durum vardır. Örneğin:

Petrol yataklarından rafinerilere boru hattıyla bağlanmış şebekenin minimum maliyetinin akış çizelgesinin belirlenmesi.
İnşaat projesinin faaliyetleri için zaman çizelgesinin (başlangıç ver bitiş) belirlenmesi.
Var olan yol ağında iki şehir arasındaki en kısa rotanın belirlenmesi.

Bu durumların çözümü çeşitli şebeke optimizasyon algoritmalarıyla gerçekleştirilir. Bir şebeke bağlantılar ile birbirine bağlanmış bir dizi düğümden oluşur. Gözönüne aldığımız bir şebeke (N,A) notasyonu ile ifade edilir. Burada N; düğümler kümesi, A ise bağlantılar kümesidir. Örneği şekil 2.1. de gösterilmiştir [6-10].

N:{1,2,3,4,5}
A:{(1,3),(1,2),(2,3),(2,4),(2,5),(3,4),(3,5),(4,5)} olmak üzere



Şekil 2.1. Bir Şebeke Ağı

Problemlerin çözümü için en çok kullanılan iki şebeke algoritması vardır. Bu algoritmalar aşağıda problem gösterimleriyle örneklenmişlerdir.

Minimum maliyet kapasiteli şebeke algoritması

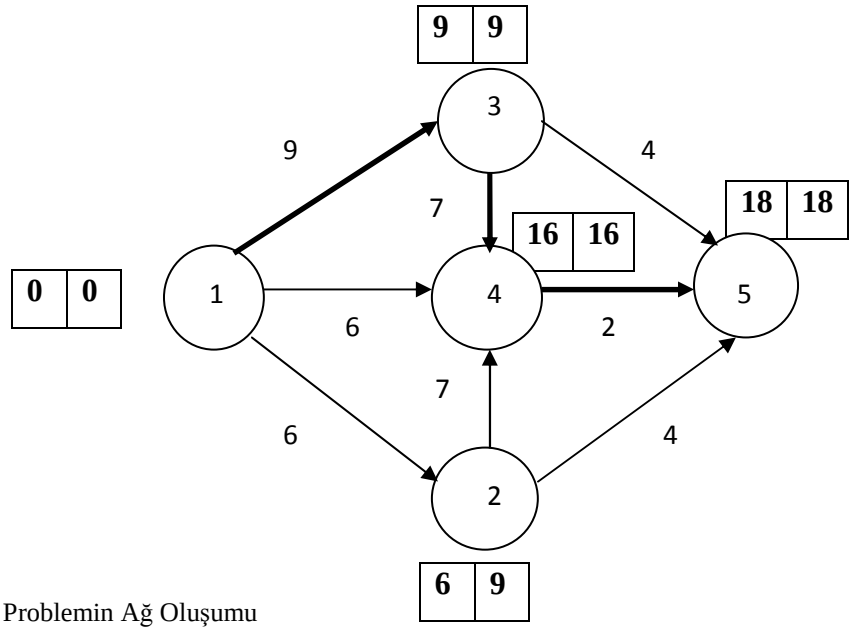
Örnek: Bir bölgeye doğalgaz bağlantı projesinde faaliyetler, süre ve indirgenmiş maliyetler aşağıdaki gibidir [8].

Tablo 2.11. Probleme ait faaliyet, zaman, normal ve indirgenmiş maliyetler

FAALİYETLER	**NORMAL**		**İNDİRGENMİŞ**	
	Süre(hafta)	Maliyet(TL)	Süre(hafta)	Maliyet(TL)
1-2	6	1000	5	2000
1-3	9	2000	7	4000
1-4	6	4000	4	6000
2-4	7	3000	6	5000
3-4	7	1000	6	3000
4-5	2	5000	1	6000
2-5	4	8000	3	9000
3-5	4	6000	3	7000

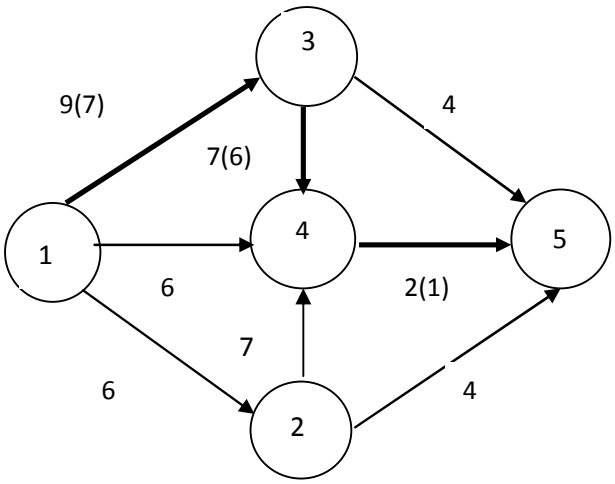
Projeyi mümkün olan en kısa sürede bitirmenin optimal maliyetini bulunuz.

Çözüm



Şekil 2.2. Problemin Ağ Oluşumu

Kritik yol: 1-3-4-5 Normal Maliyet: 30000 TL



Şekil 2.3. Problemin Ağ Oluşumu ve Kritik Yolu

İş indirgenerek 14 (7+6+1) günde bitirilebilir. En kısa sürede bitirmenin optimal maliyeti ise;

Optimal Maliyet = Normal Maliyet + Fark Maliyeti (1000+2000+1000) = 35000 TL

Tablo 2.12. Problemin Normal ve İndirgenmiş Maliyetleri

Kritik Faaliyetler	**Normal Maliyet (TL)**	**İndirgenmiş Maliyet (TL)**	**Fark Maliyeti (TL)**
1-3	2000	4000	$\frac{4000-2000}{9-7}=1000$
3-4	1000	3000	$\frac{3000-1000}{7-6}=2000$
4-5	5000	6000	$\frac{6000-5000}{2-1}=1000$

Dijkstra algoritması ile en kısa yolların bulunması

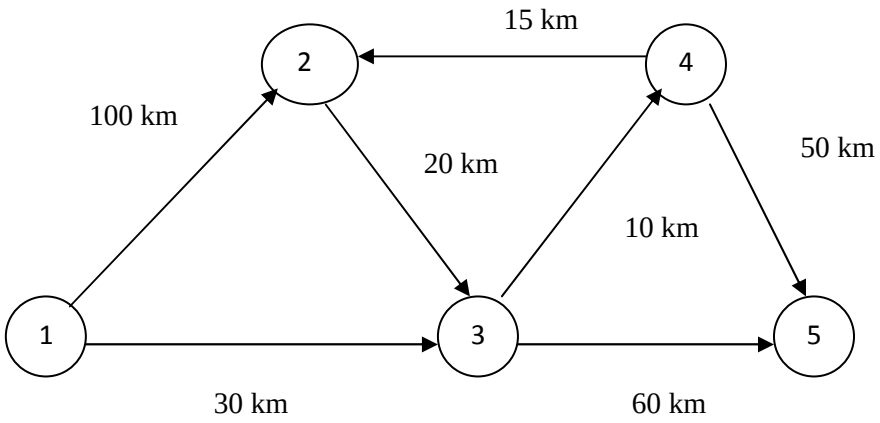
Dijkstra algoritması, kaynak düğümüyle ağdaki başka bir düğüm arasındaki en kısa yolu belirlemek üzere tasarlanmıştır. Algoritma bir etiketleme prosedürü kullanır. Etiketleme şu şekilde yapılmaktadır:

u_i 1**.**düğümden i**.** düğüme en kısa uzaklık,

$d_{ij} \geq 0$ ve (i,j) bağlantısının uzunluğu olmak üzere;

Düğüm etiketleri geçici ve kalıcı olarak işaretlenirler. Geçici etiket, aynı düğüme daha kısa bir yol bulunursa başka bir etiketle değiştirilir. Daha iyi bir yol bulunamayacaksa etiket kalıcı olarak işaretlenir. Algoritma adım adım aşağıdaki örnek ile açıklanabilir.

Örnek: Aşağıdaki şekil 2.4. te bir şehir içi karayolu ağı verilmiştir. Bu ağı kullanarak merkezler arasındaki en kısa yolları Dijkstra algoritmasını kullanarak bulun. Kutu içindeki sayılar mesafeleri göstermektedir [6-7].



Şekil 2.4. Problemin Karayolu Ağı

Çözüm

Adımlar halinde çözüm gerçekleştirilecektir.

0.adım: 1.düğüme [0,-] kalıcı etiketi atanır.

1.adım: 2. ve 3. düğümlere 1.düğümden (en son kalıcı etiketlenen) ulaşılır ve düğümler, aşağıda gösterilen Tablo 2.13. de ki gibi etiketlenir.

Tablo 2.13. Adım1 Düğümler ve Etiketleme

düğüm	etiket	statü
1	[0, -]	kalıcı
2	[0+100,1]=[100,1]	geçici
3	[0+30,1]=[30,1]	geçici

3.düğüm en kısa yolu verdiği için bir sonraki tabloda statüsü kalıcı olarak işaretlenecektir.

2. adım: 4. ve 5. düğümlere 3.düğümden ulaşılmaktadır ve yeni etiketleme aşağıda tablo 2.14. de ki gibidir.

Tablo 2.14. Adım2 Düğümler ve Etiketleme

düğüm	etiket	statü
1	[0, -]	kalıcı
2	[100,1]	geçici
3	[30,1]	kalıcı
4	[30+10,3]=[40,3]	geçici
5	[30+60,3]=[90,3]	geçici

3.adım: 2. ve 5. düğümlere 4.düğümden ulaşılabilir. 4. düğümün statüsü değiştirilir.

Tablo 2.15. Adım3 Düğümler ve Etiketleme

düğüm	etiket	statü
1	[0, -]	kalıcı
2	[40+15,4]=[55,4]	geçici
3	[30,1]	kalıcı
4	[40,3]	kalıcı
5	[90,3]veya [40+50,4]=[90,4]	geçici

2.düğümün etiketi daha kısa yol bulunduğundan değiştirilir.

4.adım: 2. düğümden sadece 3. düğüme gidilebilir. 3.düğümün etiketi kalıcı olduğu için yeniden etiketlenemez. 2.düğümdeki etiket de kalıcı olarak işaretlenir. 5. düğümden diğer düğümlere gidiş olmadığı için işlem tamamlanır.

1.düğüm ile ağdaki başka bir düğüm arasındaki en kısa yolu bulmak için, istenen varış düğümünden başlanır ve kalıcı etiketler kullanılarak geriye doğru gidilir. Örneğin 1.düğümden 2. düğüme giden en kısa yol şekil 2.5. te ki gibidir;

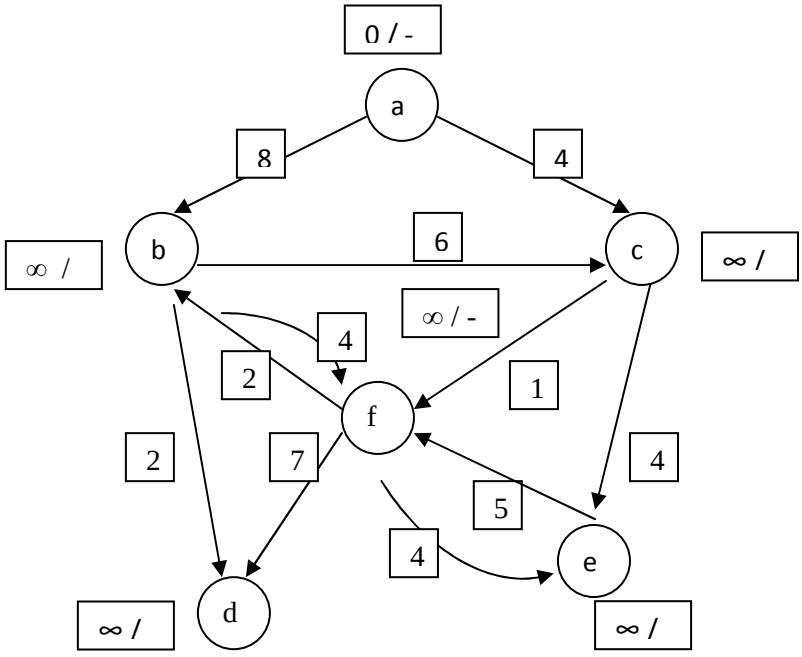
(2) [55,4] ⟶ (4) [40,3] ⟶ (3) [30,1] ⟶ (1)

Şekil 2.5. 1.düğümden 2.düğüme giden en kısa yol

Aranan yolun uzunluğu 1→ 3→ 4→ 2 şeklinde olup toplam mesafe 55 km dir.

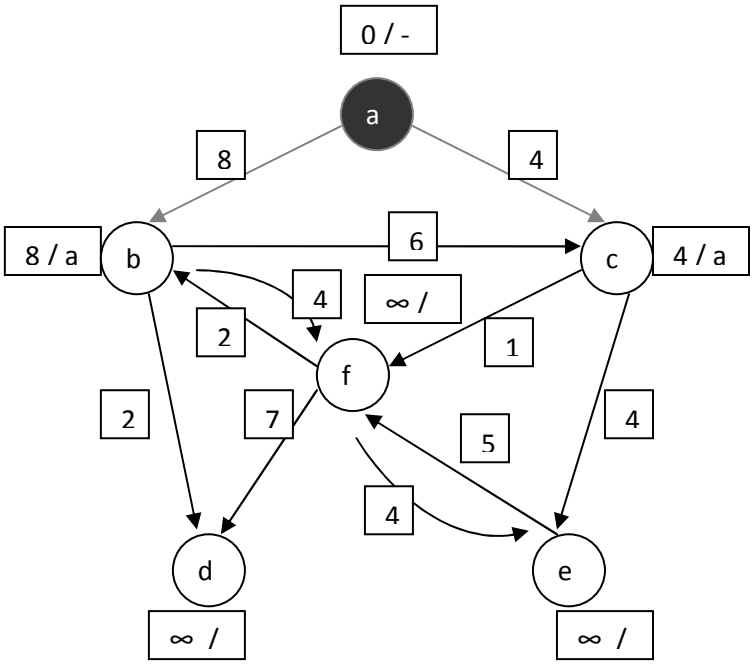
Aynı algoritmayı ağda yer alan düğümleri boyayarak uygulanması gösterilsin.

Başlangıç: a düğümünden başlanıyor, a düğümü [0/-] diğerleri [∞/-] olarak etiketlenir.



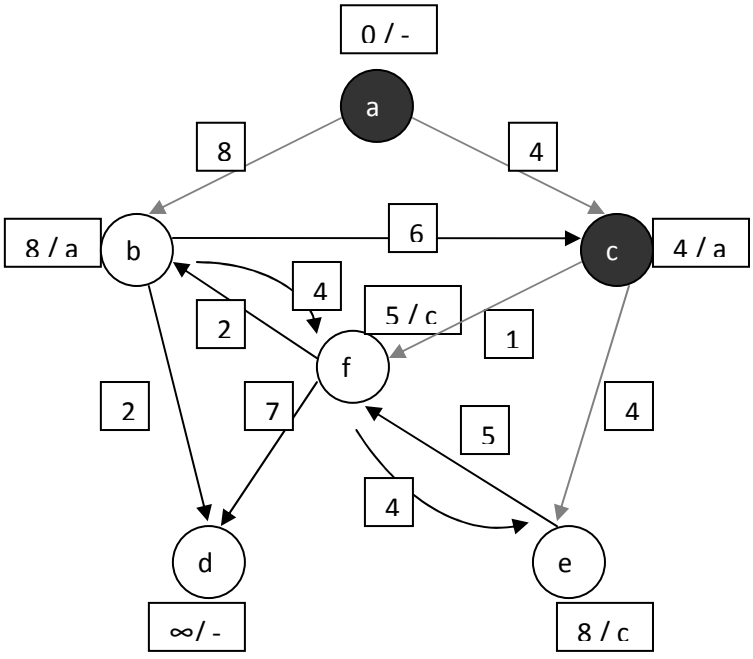
Şekil 2.6. Adım Gösterimi

0.adım: a düğümü seçilip boyanıyor, bu düğümden çıkan dallar da renklendiriliyor.



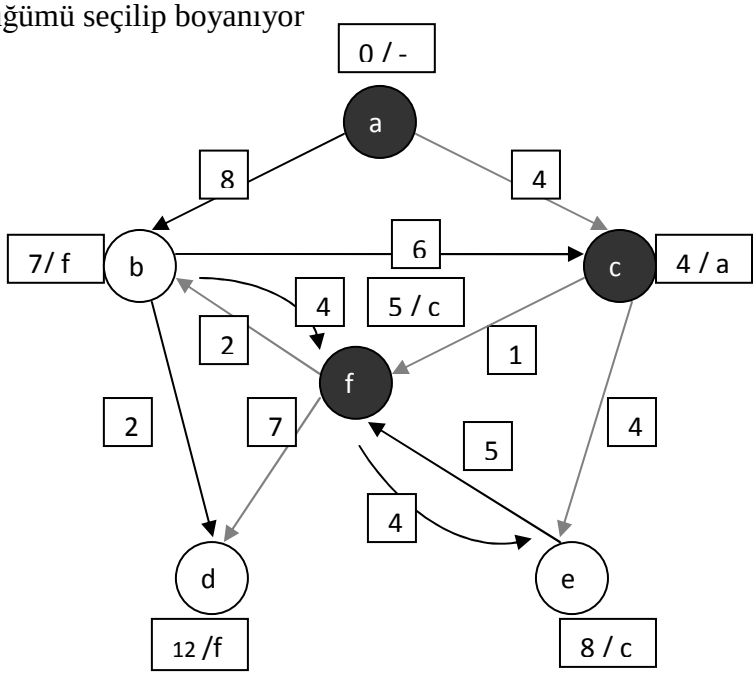
Şekil 2.7. Adım 0' ın Gösterimi

1.adım: minimum mesafe seçiliyor, seçilen düğüm boyanıyor



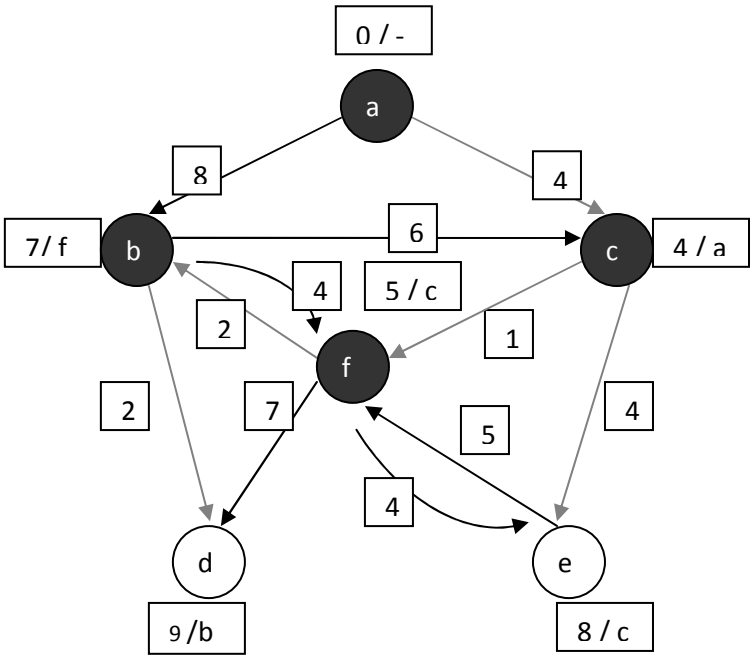
Şekil 2.8. Adım1' in Gösterimi

2.adım: f düğümü seçilip boyanıyor



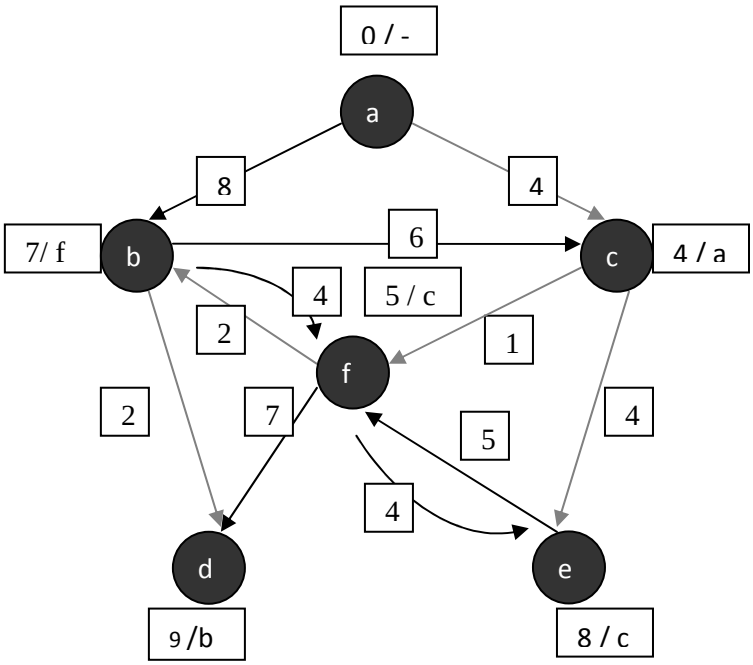
Şekil 2.9. Adım2' nin Gösterimi

3.adım: b düğümü seçilerek devam ediliyor



Şekil 2.10. Adım 3' ün Gösterimi

4.adım: d düğümü seçiliyor ve en kısa yol bulunarak **e** düğümü de boyanıyor.



Şekil 2.11. Adım 4'ün Gösterimi

- **Tamsayılı optimizasyon**

Bu optimizasyon yöntemi değişkenlerden bazılarının veya tümünün tamsayılı (kesikli) değerler aldığı problemlerin çözümünde kullanılır. Burada iki önemli tam sayılı optimizasyon algoritmasından bahsedilecektir.

Dal sınır algoritması

Bu algoritma başarılı sonuçlar veren hesap yöntemlerinden yararlanma üzerine inşa edilmiştir. Bu algoritma aşağıdaki adımları içermektedir [11].

Adım1. Programlama modelini çözerek sürekli optimumu belirlemek.

Adım2. Sürekli optimumdan başlayıp, tekrarlı bir şekilde özet kısıtlar ekleyerek çözüm uzayında düzeltmeler yapmak, böylelikle tamsayılı gereksinimleri karşılayacak şekilde bir optimum uç noktaya ulaşmayı sağlamak.

Örnek: Bir gsm firma bayisi kullanıcılarına iki ayrı opsiyonla hizmet vermektedir. Birinci opsiyonlu günlük satışların her biri 8TL, ikinci opsiyonlu günlük satışların her biri 5TL lik bir kâr getirmektedir. Günlük satış opsiyon limiti 6 adet ile sınırlıdır. Birinci opsiyonda 9 TL, ikinci opsiyonda 5TL olmak üzere bir kullanım ücreti alınmakta, bu ücret iki opsiyon için 45 TL ile sınırlandırılmıştır. Bayi kârını maksimum yapmak için nasıl bir satış politikası izlemelidir.

Çözüm

x_1: Birinci opsiyondaki günlük satış adedi

$x_{2.}$ İkinci opsiyondaki günlük satışa adedi

$Zmax = 8x_1 + 5x_2$

$x_1 + x_2 \leq 6$

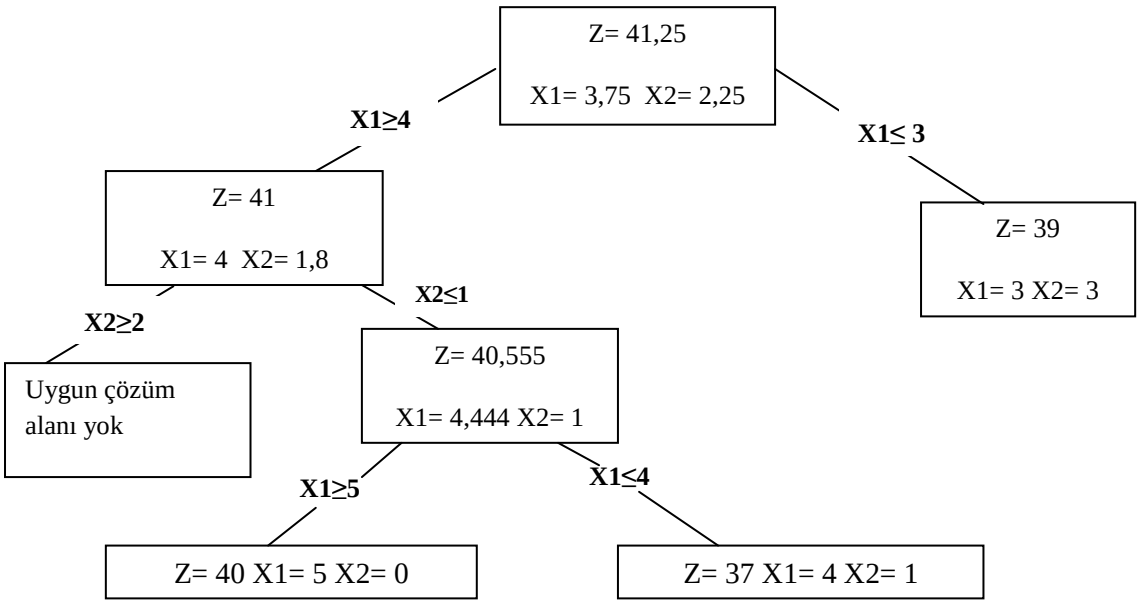
$9x_1 + 5x_2 \leq 45$

$x_1 \geq 0$

$x_2 \geq 0$

Oluşturularak problem çözüldüğünde

Z = 41,25 $x_1 = 3{,}75$ $x_2 = 2{,}25$ olarak bulunur. Burada x_1, x_2 tamsayı olmalıdır o nedenle problemin dal sınır algoritması ile çözümü şu şekildedir.



Şekil 2.12. Problemin Dal Sınır Algoritması Gösterimi

Çözüm Z = 40 $x_1 = 5$, $x_2 = 0$ da sağlanmış oldu.

Gezgin satıcı problemi çözüm algoritması

Örnek: X şehrinde acil ambulans servisine gereksinim olan altı bölge bulunmaktadır. Ambulans merkezleri bu altı bölgenin hepsinde yer alabileceği gibi, bazılarında da yer alabilir. Bunun yanında bazı bölgelerin birbirine yakınlığı nedeniyle birden çok bölgeye tek bir merkezin hizmet vermesi söz konusu olabilmektedir. Burada sadece merkezlerin 15'er dakikalık araba mesafesi içinde olma zorunluluğu vardır. Tablo 2.16 da bölgeler arası sürüş mesafeleri (zaman) verilmektedir [4].

Tablo 2.16. Bölgeler Arası Sürüş Mesafeleri

Merkezler	**1**	**2**	**3**	**4**	**5**	**6**
1	0	10	20	30	30	20
2	10	0	25	35	20	10
3	20	25	0	15	30	20
4	30	35	15	0	15	25
5	30	20	30	15	0	14
6	20	10	20	25	14	0

Buna göre en az sayıdaki merkezi ve konuşlanacağı yeri bulacak şekilde problemi formüle edip Lindo alt modülünde çözümün yapılması.

Çözüm

X_i: i. merkezde konuşlanma durumu

$Zmin = X_1 + X_2 + X_3 + X_4 + X_5 + X_6$

$X_1 + X_2 \geq 1$

$X_1 + X_2 + X_6 \geq 1$

$X_3 + X_4 \geq 1$

$X_3 + X_4 + X_5 \geq 1$

$X_4 + X_5 + X_6 \geq 1$

$X_2 + X_5 + X_6 \geq 1$

Çözümün Lindo alt modülüne uygulanması

min $X_1 + X_2 + X_3 + X_4 + X_5 + X_6$

st

$X_1 + X_2$ >=1

$X_1 + X_2 + X_6$>=1

$X_3 + X_4$>=1

$X_3 + X_4 + X_5$>=1

$X_4 + X_5 + X_6$>=1

$X_2 + X_5 + X_6$>=1

End

INTE X_1

INTE X_2

INTE X_3

INTE X_4

INTE X_5

INTE X_6

Çözüm sonucu

$X_2 = 1, X_4 = 1,$ Zmin = 2 olarak bulunur. (Ambulansların 2. ve 4. merkezlere konuşlanması yeterlidir.)

Örnek: Bir firma metalleri işlemek için matkap torna ve freze tezgahlarına sahiptir. Firmada çalışan işçilerin tezgahlardaki çalışma süreleri aşağıdaki gibidir [14].

Tablo 2.17. Tezgahlar ve Çalışma Süreleri

İşçiler	Matkap	Torna	Freze
1	22 dk	18 dk	35 dk
2	41 dk	30 dk	28 dk
3	25 dk	36 dk	18 dk

Firma üretim zamanını en az şekilde gerçekleştirmek için hangi işçiyi hangi tezgaha nasıl atamalıdır.

Çözüm

X_{ij}: i. işçinin j. tezgahta çalışma durumu

Zmin = $22X_{1M} + 18X_{1T} + 35X_{1F} + 41X_{2M} + 30X_{2T} + 28X_{2F} + 25X_{3M} + 36X_{3T} + 18X_{3F}$

$X_{1M} + X_{1T} + X_{1F} = 1$

$X_{2M} + X_{2T} + X_{2F} = 1$

$X_{3M} + X_{3T} + X_{3F} = 1$

$X_{1M} + X_{2M} + X_{3M} = 1$

$X_{1T} + X_{2T} + X_{3T} = 1$

$X_{1F} + X_{2F} + X_{3F} = 1$

Çözümün Lindo alt modülüne uygulanması

min $22X_{1M} + 18X_{1T} + 35X_{1F} + 41X_{2M} + 30X_{2T} + 28X_{2F} + 25X_{3M} + 36X_{3T} + 18X_{3F}$

st

$X_{1M} + X_{1T} + X_{1F} = 1$

$X_{2M} + X_{2T} + X_{2F} = 1$

$X_{3M} + X_{3T} + X_{3F} = 1$

$X_{1M} + X_{2M} + X_{3M} = 1$

$X_{1T} + X_{2T} + X_{3T} = 1$

$X_{1F} + X_{2F} + X_{3F} = 1$

End

INTE X_{1M}

INTE X_{1T}

INTE X_{1F}

INTE X_{2M}

INTE X_{2T}

INTE X_{2F}

INTE X_{3M}

INTE X_{3T}

INTE X_{3F}

Sonuç $X_{1M} = 1,\ X_{2T} = 1,\ X_{3F} = 1$ ve Zmin = 70 dakika olarak bulunur.

- **Dinamik programlama optimizasyonu**

Dinamik programlama, ilk olarak 1957 yılında Richard Bellman tarafından ortaya konmuş bir optimizasyon yöntemidir. Optimizasyon seçilen pek çok optimallik ölçüleri içinde en olumlu ve kararlı olan yolun bulunmasıdır. Optimallik ölçüleri performans indeksleri olarak adlandırılır. Optimizasyon için minimize veya maksimize edilecek performans indekslerine ihtiyaç duyulur. Bellman dinamik programlama yöntemini şu şekilde tanımlamaktadır: “Bir optimal politika kararı, başlangıç durumu ve ilk karar ne olursa olsun daha sonra verilecek olan kararlar ilk verilen kararların sonucunda ulaşılan durum göz önüne alınarak verilmelidir” [12].

Bir problemin çözümüne genellikle, uygun bir matematiksel modelinin oluşturulmasıyla başlanılır. Modelin kurulması aşamasında amaç fonksiyonu ve kısıt koşulların seçilmesi, değişkenlerin sayı ve özellikleri ve yapılacak varsayımlar büyük ölçüde model kurucunun bilgi ve deneyimlerine bağlı olmaktadır. Bu bağımlılık modelin çözümü aşamasında da geçerlidir. Problem modelinin çözümlenmesine başlanmadan önce genellikle problemin yeniden düzenlenmesine ve uygulanacak çözüm yöntemine uyarlanması veya birtakım değişken dönüşümü işlemlerinin yapılmasına gerek duyulmaktadır. Dinamik programlama yönteminde çözümlenmesi planlanan problemler bir bütün olarak çözümlenmek yerine, daha az sayıda değişkenler içeren küçük problemlere ayrıştırılarak çözülürler. Bu şekilde çok aşamalı bir karar süreci, tek aşamalı problemler dizisine dönüştürülebilir. Bir başka deyişle optimize edilmesi istenen bir problem, aşamalı olarak optimize edilecek küçük ve tek değişkenli problemlere dönüştürülür [12-13].

Dinamik programlama, çözümü güç olabilecek büyük bir problemi daha küçük alt problemlere ayrıştırarak, çözümde kolaylık sağlayan bir optimizasyon tekniği olarak tanımlanabilir. Bir problemin dinamik programlama yöntemiyle çözümlenebilmesi için söz konusu problemin, biri diğeri ile bağlantılı alt problemlere ayrıştırılabilme özelliğine sahip olması gerekir. Böyle bir problem için geliştirilecek karar modeli, bütüne bağlı (alt) karar modelleri biçiminde ele alınabilir. Her bir alt problem ayrı ayrı incelenir ve problemin tamamı optimal olacak şekilde çözümlenir. Dinamik programlama, matematiksel programlama problemlerini daha basit, birbirinden

bağımsız alt problemlere ayrıştırdıktan sonra aşama aşama bu alt problemlere çözümler getiren ve sonuçta orijinal problem için optimal bir sonuç ortaya koyan bir yaklaşımdır.

Dinamik programlama yöntemi, birbiri ile ilişkili kararlar serisinin çözümünde kullanılan bir sayısal yöntemdir. Koşulların zaman süreci içinde değiştiği ve bu koşulların verilecek kararlar üzerinde önemli etkilerinin olduğu biliniyorsa, dinamik programlama modellerine ihtiyaç duyulur. Dinamik programlamada, mevcut sistem birbiri ardından işlem gören parçalara ayrılmakta ve ardışık iki işlem arasında fonksiyonel bir bağıntı kurulması yoluna gidilmektedir. Bu yöntem daha çok birbirleriyle ilişkili bir dizi kararlar almayı gerektirdiğinden işlemler yineleme denklemleri kullanılarak yapılmaktadır. Yineleme denklemleri ile optimizasyonda, optimizasyon bir önceki kararın içerdiği bilgilerden yararlanarak adım adım gerçekleştirilir. Her adımda bulunan çözüm kendi basına problemin çözümü olmayıp, optimal çözümün bir parçasını belirleyen bilgiyi içermektedir. Her alt problemde verilen karar bir sonraki aşamada verilecek olan kararı etkileyeceğinden, her alt problemde verilen kararın sadece o alt probleme olan etkileri değil aynı zamanda sonraki bütün alt problemlere olan etkileri de göz önüne alınmalıdır. Alt problemler birbirleriyle bağlantılı olduklarından bir sonraki alt problem için gereken bilgi bir önceki alt problemden elde edilen bilgi olacaktır.

Dinamik programlamanın çözüm yöntemi olarak kullanıldığı işletme problemleri arasında şu problemleri sayabiliriz; sermaye bütçeleme problemi, fiyat stratejisi belirleme problemi, kargo yükleme problemi, dağıtım problemi, pazarlama ve yatırım problemi, üretim planlama problemi, envanter problemi.

Dinamik programlamanın en fazla uygulandığı işletme sorunlarını da aşağıdaki biçimde sıralayabiliriz.

Yeniden sipariş kurallarının belirlenmesinde zaman ve nicelik değişkenlerinin saptanması,
Değişen işlem koşullarında üretim programlaması ve işgücü düzenlemesi,

Pahalı araç ve gerecin etkin bir biçimde kullanılmasını sağlamak üzere yedek parça düzeyinin belirlenmesi,
Yeni alanlara kaynak dağıtımı yapan sermaye bütçelemesi,
Ürünleri halka geniş ölçüde tanıtabilmek için reklam araçlarının seçimi,
Değerli bir kaynağın bulunmasında sistematik aramanın yapılması,
Karmaşık makinelerin bakım onarımının programlanması,
Eskiyen donanım ve makinelerin yenilenmesi için uzun dönem stratejilerinin saptanması,
Çeşitli malzemeler için kesme kalıplarının belirlenmesi,
Endüstride kullanılan robot gruplarının plan sırasını, görev ve denetiminin optimal performansını sağlayan yörüngelerin belirlenmesi

Dinamik programlama kavramları

Dinamik programlama matematiksel programlama problemlerini daha basit, birbirinden bağımsız alt problemlere ayrıştırdıktan sonra aşama aşama bu alt problemlere çözümler getiren ve sonuçta orijinal problem için optimal bir sonuç ortaya çıkartan bir yaklaşımdır [12].

Dinamik programlama yöntemi birkaç kavram üzerine kuruludur. Bu kavramların bazıları diğer modellerle aynı, bazıları ise yalnızca dinamik programlama yöntemine aittir. Dinamik programlamada en önemli olgu “değişken” olgusudur. Sistem içinde değişkenlerin belirlenmesi yanında bunların karmaşıklığının en aza indirilmesi ve basitleştirilmesi de gerekmektedir. Karmaşık değişkenler modelin çözümünü güçleştirmektedir. Sistem hakkında karar verilmesi gerekmekte, sistemin her aşamasında karar alınmakta, sonraki aşamalarda bu kararlar kullanılarak farklı kararlar alınmakta ve amaca ulaşılmaya çalışılmaktadır. Alınacak olumlu kararların sonucunda bir takım ödül ve kazançlar ortaya çıkmakta ve bunlar sonraki kararların olumlu sonuçlar doğurmasına neden olmaktadır.

Verilen kararları karşılaştırarak içlerinden maksimum seviyede faydalı olanı seçmek en iyi yöntemdir. Eğer değişmeyen n kısmın değeri hesaplanabilirse, n+1 değişmeyen kısmında hesabı yapılabilmektedir. Ardışık karar problemi sonuç

çıkarma kavramlarına ait değerlerin zeka yoluyla sağlanmasıyla çözülmektedir. Bu çözüm dinamik programlama yöntemi olarak adlandırılmaktadır. Dinamik programlama yönteminde sürekli olarak kullanılan sembol ve terimleri açıklamak, problemleri daha kolay formüle etmek ve çözmek için yararlı olacaktır.

Aşama

Büyük bir problem çözümlenmek üzere daha küçük alt problemlere ayrıştırıldığında her alt problem çözümlenmesi gereken bir karar problemi olarak düşünülürse, karar verilmesi gereken her nokta aşama olarak tanımlanır. Bu bağlamda her alt problem bir aşamaya karşılık gelmektedir.

Dinamik programlama bir durumdan başka bir duruma geçişte, seri hareketlerin içine alındığı süreçle ilgili bir sistemdir. Aşama bir sürecin içindeki tek bir adımdır ve aynı özellikli durumlardan komşu durumlara geçişi ifade eder. Aşama zaman faktörü olabileceği gibi daha farklı faktörlerde olabilir.

Örneğin maliyet minimizasyonunu amaçlayan yıllık stok planlama modelinde, stok miktarları aylık olarak belirlenirse her ay bir aşama olur. Bir gemiye her yükün farklı bir gelir sağladığı n çeşit yükün yüklenmesi probleminde, kârı maksimize etmek amacıyla hangi yükten ne kadar yüklenmesi gerektiği bulunmaya çalışılırken, her çeşit yük bir aşamaya karşılık gelir.

Aşamalar birbirleriyle bağlantılı kararlar serisi oluşturmaktadırlar. Bu nedenle her aşamada alınan karar yalnız bir sonraki aşamayı değil problemin sonuna kadar bütün aşamaları etkilemektedir.

Durum

Herhangi bir aşamadaki durum, daha önceki aşamalarda verilen kararların sonucu olarak tanımlanabilir. Bir aşamada verilen karar, bu aşamadaki değişkenler kümesini yeni bir değişkenler kümesine çevirmektedir. Yeni değişkenler kümesine uygun

olarak durumda değişecektir. Durum, optimal bir karar vermek için herhangi bir aşamada ihtiyaç duyulan bilgi olarak da düşünülebilir.

Durum kavramını, her bir aşamada sistemin veya değişkenlerin alabileceği değer, başka bir ifadeyle, bir aşama ve onu izleyen aşamalara dağıtılan kaynaklar şeklinde tanımlamak da mümkündür. Herhangi bir aşamadaki süreci koşulu olarak da tanımlanan durum kavramı ile incelenen konunun içerdiği tüm bilgiler ve sınırlamalar anlatılır. Sistemin verilen bir durumuna bağlı olarak ortaya çıkan ilgili aşamadaki eylem seçeneklerine durum değişkenleri adı verilir. Durum kavramı mutlak bir kavram olmayıp, analizin özelliğine bağlıdır. Her hangi bir stok problemi için stok düzeyi, üretim problemi için üretim düzeyi, her hangi bir aşamanın durumunu gösterebilir.

Durum değeri

Durum değeri, sistem mevcut durumdan başlayıp özel bir politika izlediğinde ortaya çıkan getirilerin toplam bir fonksiyonudur. Optimal politika uygulandığında durum değeri optimal değerdir.

Hareket

Karar değişkeni olarak da ifade edilebilen hareketler, dinamik programlama modelinin her bir aşamasındaki olası kararları belirtmektedir. Her bir durumdan başka durumlara yapılması gereken birçok olanaklı hareket vardır. Dinamik programlama problemi çözümü istenilen amaca göre en iyi hareket dizisinin seçimini içermektedir.

Karar

Dinamik programlama her hangi bir problem için mümkün olan koşullar içinde ne yapılacağına karar vermeyi sağlar. Karar, problemdeki bir aşamayı tamamlamak için ilgili alternatifler arasından bir seçim yapılması işlemidir. Belirli bir durum ve aşamada verilen bir karar, sürecin hem durumunu hem de aşamasını değiştirir.

Dolayısıyla her karar, geçerli bir durumdan bir sonraki aşamaya bağlı olan duruma geçişi etkilemektedir. Çok aşamalı bir karar süreci aşamalara ayrıldıktan sonra her aşama için bir dönüşüm fonksiyonu oluşturulur. Her aşamada karar verme süreci, o aşamanın seçeneklerinden birinin seçilmesiyle sonuçlanır. Buna asama kararı denir. Ayrıca her hangi bir aşamadaki karar optimal ise bu karar, problemin optimal çözümünün bir parçası olarak kullanılabilir.

Getiri

Her karar amaç fonksiyonuna katkı sağlayan bir getiri ile sonuçlanır. Getiri bir sürecin her bir aşamasında meydana getirilen bir sistemdir. Getiri genellikle kazanç, maliyet, mamul üretimi ya da kaynak tüketimi olabilmektedir.

Ardışık en iyileme

Dinamik programlama yönteminde problem aşamalara ayrıştırıldıktan sonra her aşamada belirli bir optimizasyon amacına bağlı olarak kararlar verilmektedir. Bir aşamada verilen karar tek başına tüm problemin optimum çözümünü temsil etmez. Problemin tamamına ait optimum çözümü elde etmek için aşama sonuçları birleştirilmektedir. Aşamalar arasındaki bağlantıyı iç içe fonksiyonlar sağlamaktadır. Bu fonksiyonların kullanılması sonucu her bir aşamada bir önceki aşama sonuçları birleşerek en iyi çözümü veren ardışık kararlar dizisi elde edilmiş olur.

Dönüşüm fonksiyonları

Her aşamanın bulunabilecek durumlarında verilebilecek karara göre bu aşamayı izleyen veya daha önceki aşamanın hangi durumuna gelineceğini belirleyen ilişkilere dönüşüm fonksiyonları denilmektedir. Dönüşüm fonksiyonları çözüm yolunun ileriye veya geriye doğru en iyileme olmasına göre farklılık gösterir.

Optimal politika

Çok aşamalı bir karar sürecinin her karara bağlı, maliyet ve kâr cinsinden bir sonucu vardır. Bu sonuç sürecin aşama ve durumuyla birlikte değişir. Karar sürecinin tüm aşamalarını kapsayan belirli seçenekler dizisi politika, karar vericinin amacını gerçekleştirecek optimum seçenekler dizisi de optimum politika olarak adlandırılır. Optimal politika sürecin her bir aşaması için verilen kararların bir sırasıdır. Çözüm sıra önceliğine göre bir aşamadan diğerine gidilerek elde edilir ve son aşamaya erişildikten sonra her parametre için değerler belirlenerek işlem tamamlanır. Böylece en uygun politika oluşturulmuş olur.

Dinamik programlama problemlerinin özellikleri

Büyük ve çok sayıda karar değişkeni olan sorunları ardışık küçük sorunlara bölerek çözmek için geliştirilmiş olan dinamik programlama yönteminin temel özellikleri aşağıda açıklanmıştır [12-14].

Problemler, belirli seçenekleri veya karar politikalarını içeren bir dizi sıralı aşamalara bölünebilir. Aşamalar mekan yönünden farklı yer ve yolları, zaman boyutunda ise farklı noktaları gösterebilir. Tüm dinamik programlama problemleri sıralı karar aşamalarına ayrılarak, karar problemlerinin bir aşamaya karşılık geldiği birbiriyle ilişkili kararlar dizisinin oluşturulmasını gerektirmektedir.

Her bir aşama o aşamanın başlangıcıyla ilgili bir araya gelmiş birçok duruma sahiptir. Durumlar problemin belirtilen aşamasında sistemin içinde bulunduğu olası koşulardır. Durumların sayısı sonlu veya sonsuz olabilmektedir.

Karar politikasının sonucunda her bir aşamada bir araya gelmiş durumların birinden, gelecek aşamadaki bir duruma ulaşılır. Bir diğer ifadeyle her bir aşamadaki mevcut durum ile politika kararı, bir sonraki aşamanın başlangıcına ilişkin bir duruma dönüşür.

Çözüm yönteminde optimal bir politikanın bulunması her bir aşamada optimal politika kararlarının verildiği tüm problemler için modellenebilir.

Bir optimal politika kararının verildiği mevcut durumda, müteakip aşamalarda alınacak kararlar daha önceki aşamalarda alınan kararlardan bağımsızdır. Bu bağımsızlık durumu, dinamik programlama için optimalite prensibi olarak bilinmektedir.

Çözüm yöntemine son aşama için optimal politikanın bulunması ile başlanır. Son aşama için optimal politika, o aşamada ki her bir olası durum için optimal politika kararını içermektedir.

Aşama n+1 için optimal politika belli ise aşama n için optimal politikayı belirten tekrarlı yapıya ulaşılabilmektedir. Asama n'de ve s durumunda başlanılması halinde optimal politika kararının verilmesi x_n' in değerinin bulunmasını gerektirmektedir. Dinamik programlama karar probleminin fonksiyonel eşitlik şeklindeki modeli şu şekilde gösterilir.

$f^*_n(s_n) = \max [f_n(s_n, x_n)]$
veya
$f^*_n(s_n) = \min [f_n(s_n, x_n)]$

Problemin çözümünde, yineleme ilişkisi kullanılarak çözüm işlemleri sondan başa veya baştan sona doğru aşama aşama sürdürülür. Her aşamada, her bir durum için optimal politika bulunarak, ilk veya son aşamadaki optimal politika elde edilinceye kadar işlemlere devam edilir. Problem çözümlenirken aşamalarda geriye veya ileriye doğru gidilebilinirse de, özellikle aşamaların zaman boyutundaki noktalara denk olduğu problemlerde, işlemler sondan başa doğru yapılır ve değerlendirilen bir aşamaya yeniden dönülmez.

Dinamik optimizasyon tekniğinin özellikleri özetle şu şekilde ifade edilebilir:

Çok aşamalı problem alt parçalara, adımlara veya tek tek aşamalara bölünür. Buna "ayrıştırma" veya "dekompozisyon" denir. Aşamalarda her seferinde bir kez olmak üzere (veya yineleme ile), belirli bir optimizasyon amacına bağlı kalarak kararlar verilir. Problemin tümüne ait çözümü elde etmek için aşama sonuçları birleştirilir. Birleştirme sonucunda politika olarak adlandırılan ardışık kararlar dizisi elde edilir.

Dinamik programlamanın avantaj ve dezavantajları

Dinamik programlama, sırayla verilen kararlarla ilgili bir tekniktir ve alınan her bir karar bir sonraki kararı etkileyecek sonucu ortaya çıkartmaktadır. Dolayısıyla en son aşamada, önceden ortaya konulan optimal değerler hesaba katıldığından her aşama aynı yöntemle optimumlaştırılmalı ve sonraki kararlar önceden verilen kararlar zincirinin üzerindeki etkisini göz önünde bulundurmalıdır. Bu düşünüş felsefesi ile dinamik programlamanın bir takım üstünlükleri ve zayıf yönleri bulunmaktadır. Bunlar aşağıda dinamik programlamanın avantaj ve dezavantajları olarak açıklanmıştır.

Dinamik programlamanın avantajları

a) Zor ve karmaşık problemler, dinamik programlama yaklaşımı ile birbirleriyle ilişkili alt problemlere ayrılarak daha kolay bir şekilde çözümlenebilmektedirler. Dinamik programlamanın bu avantajı özellikle üretim düzeltme ve planlama gibi ardışık kararlar içeren problemler için çok önemlidir.

b) Dinamik programlama diğer matematiksel programlama problemlerine uygulanabilen esnek yapısıyla, bir teknikten ziyade bir optimizasyon yaklaşımıdır. Amaç fonksiyonundaki her bir değişken, bir aşamada belirlenebilmektedir. Tamsayılı optimizasyon problemleri ve doğrusal olmayan tamsayılı optimizasyon problemleri dinamik programlama yaklaşımıyla çözülebilmektedir.

c) Dinamik programlama deterministik ve stokastik süreçlere uygulanabilmektedir.

d) Dinamik programlamanın hesaplama yöntemi duyarlılık analizi yapmaya elverişlidir.

e) Dinamik programlama problemlerin çözümünde olası tüm seçeneklerin dikkate alınması yerine, çok daha az hesaplama yapma kolaylığı sağlamaktadır.

Örneğin; 2 değişkenli ve her bir değişken için 10 farklı seçenek içeren bir problemin tam sayımı 10^2 = 100 işlem gerektirmekte, 4 değişken olması halinde 10^4 = 10.000 işlem gerektirmektedir. Değişken sayısı arttığında tam sayım çok büyük bir artış göstermektedir. Buna karşılık dinamik programlama, benzer problemlerin çözümünde çok daha az işlem gerektirmenin yanı sıra değişken sayısının 2'den 4'e çıktığı durumlarda asama sayısı da 2'den 4'e çıkmaktadır. Dinamik programlama problemlerinde amaç fonksiyonundaki değişken sayısı kadar aşama sayısı vardır.

f) Dinamik programlamanın hesaplama açısından yararlı olmasının nedeni her aşamada durum örnek uzayının sabit olmasıdır. Verilen bir giriş durumu ile başlayan karar değişkenlerinin sayısı çok fazla olduğunda, dinamik programlama hesaplama açısından çok avantajlıdır. Her aşamadaki durum sayısı azaldıkça dinamik programlamanın hesaplama avantajı azalır. Herhangi bir aşamada verilen kararlar sonraki aşamaları etkiliyorsa sayısal anlamda kolay hesaplamalar olabilir.

g) Kararları bölümlere göre ayırabilme varsayımı sayesinde tüm olası yolları tek tek hesaplamadan bazı yollar elimine edilerek optimallik analizi yapılmaktadır.

Dinamik programlamanın dezavantajları

a) Dinamik programlama diğer matematiksel programlama tekniklerinden daha zor anlaşılır kavramlara sahiptir. Problemin kavranıp uygun bir formül yardımıyla çözülebilmesi için kesinlikle bir uzmana ihtiyaç duyulmaktadır.

b) Simpleks yöntemi gibi genel bir algoritması yoktur. Bu yüzden dinamik programlama için yazılmış paket programlar çok az sayıda olmasına karşın, bu

paket programlar çoğu durumda yetersiz kalmakta veya büyük sorunlar içermektedir.

c) Dinamik programlamanın önemli bir dezavantajı boyutsallık sorunudur. Durum değişkeni ve karar değişkeninin alacağı değerlerin çok sayıda olması, her bir aşama için oluşturulacak çizelgelerin boyutunu önemli ölçüde artırmaktadır.

d) Etkin bir hesaplama algoritmasının eksikliği, dinamik programlamayı gerçek dinamik kararları almada pratik olmaktan çok kavramsal olmaya dönüştürür.

Optimallik ilkesi ve dinamik programlama formülasyonu

Dinamik programlama optimizasyonu, optimumlaştırma yöntemleri arasında bir aşama daha ileride olan tek işlem yerine birden çok işlemi sıra ile elde edebilen özel planlama ve programlama yöntemidir. Çözüm yöntemi, problemin bütününü alt problemlere bölmek ve her alt problemin optimal çözümünün bulunması yoluyla problemi çözmektir. Aşamaları oluşturan her alt problem karar sürecinin bir parçasıdır. Her aşamada optimal bir seçim yapılarak en iyi politika belirlenir. Optimallik ilkesine göre optimal politika ya da karar öyle bir özellik taşımalıdır ki bir duruma nasıl erişildiği göz önüne alınmaksızın sonraki kararlar o durumun terk edilmesinden sonra optimal bir politikayı oluşturmalıdır.

Optimallik ilkesi dinamik programlamanın esasını oluşturur. Bu ilkeye göre birbiri ardına gelen iki aşama arasında optimal değeri verebilecek biçimde ilişki kurulabilmektedir. Bellman optimallik ilkesi ile sıradan optimizasyon problemlerine önemli katkıda bulunmuştur. Problem birçok eşitlik ve olasılıklarla ifade edilebilen matematiksel modellere dayanmaktadır. Optimallik ilkesi ilk olarak çok amaçlı problemlerin çözümünde dinamik programlamaya dayanılarak başarıyla uygulanmıştır. Dinamik programlama tekniklerini ortaya koyan Bellman'ın ifadesiyle: "Optimum bir politika o şekilde olmalıdır ki; ilk durum ve ilk karar ne olursa olsun, ondan sonra verilecek kararlar, ilk karardan doğan sonuçlara rağmen optimum bir politika oluşturmalıdır. Bir başka ifadeyle izlenecek çözüm yolu, önceki kararlar ne olursa olsun aşamalarda yine optimum politikayı verir. Örneğin birinci ve

ikinci aşamada yanlış kararlar verilmiş olsa da üçüncü, dördüncü ve takip eden aşamalarda doğru kararlar verilebilir.

Dinamik programlama, her aşamadaki sonuç ve erişilmesi amaçlanan optimal sonuç arasında yinelenen ilişkiler kuran bir çözüm yaklaşımıdır. Dinamik programlamada her problem için yinelenen ilişkileri yazmak gereklidir. Bu tür bir denklem bir kez yazılınca, dinamik programlama hesaplamalarını gerçekleştirmek daha kolay olmaktadır. Her karar probleminde problemin özelliğine göre özgün bir model kurulsa dahi tüm dinamik programlama problemlerinde aşağıdaki genel formülasyona benzer formülasyonlar kullanılır.

n = Mevcut aşama (n = 1,2,3, ... , N)

n – 1 = Bir önceki aşama

s_n = n inci aşamada sistemin durumu (yinelenen ilişkileri kapsayan mevcut aşamada sistemin durumu)

s_{n-1} = n - 1 inci aşamadaki sistemin durumu

x_n = n inci aşama için karar değişkeni

x^*_n = s_n durumunda bulunan x_n' lerin optimal değeri

$r_n(s_n, x_n)$ = s_n durumundaki x_n kararı benimsendiğinde n inci aşamada gerçekleşen kazanım

$f_n(s_n)$ = n inci aşamada ki s_n durumundan başlayıp işlemlerin sonuna kadar, her alternatif için gerçeklesen toplam kazanım

$f^*_n(s_n)$ = En iyi toplam kazanım (n inci aşamadaki s_n durumunda ki en iyi $f_n(s_n)$ değeri

$f^*_{n-1}(s_{n-1})$ = n - 1 inci aşamada elde edilen en iyi toplam kazanım

Bu verilere göre yinelenen ilişkileri ele alan dönüşüm fonksiyonu, amaç minimizasyon ve çözüm ileriye doğru en iyileme olduğunda;

$f^*_n(s_n) = \min [r_n (s_n, x_n) + f_{n-1} (s_{n-1})]$ şeklinde olur.

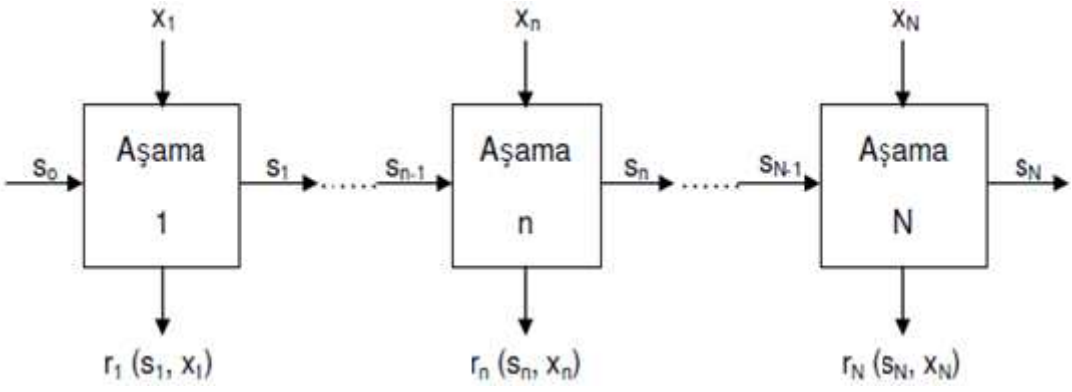
Amaç maksimizasyon ve çözüm ileriye doğru en iyileme olduğunda

$f^*_n(s_n) = \max [r_n(s_n, x_n) + f_{n-1} (s_{n-1})]$ seklinde gösterilebilir.

Dinamik programlamanın çözüm yolu ve yöntemleri

1. İleriye doğru çözüm yöntemi

Bu çözüm yönteminde n-1 inci aşama ile ilgili bilgiler, n inci aşamanın karar girdilerini oluştururlar. Buna bağlı olarak çözüme 1 inci aşamadan başlanılarak N. aşamaya doğru gidileceğinden dönüşüm fonksiyonları da buna uygun olarak formüle edilecektir [15].



Şekil 2.13. N Aşamalı Bir Problemin İleriye Doğru Çözüm Süreci

Başlangıç aşamasında sistemin durumu s_0 olarak gösterilmektedir. Birinci aşama içerisinde, birinci aşamanın çıktısını belirleyen bir dönüşüm fonksiyonu yardımıyla işlem gerçekleştirilerek x_1 karar değişkeni ile gösterilen çıktıya ulaşılır. x_1 kararının verilmesiyle elde edilen getiri ise r_1 (s_1, x_1) ile gösterilir. Süreç bu şekilde son aşamaya kadar sürdürülür. İleriye doğru çözüm yolunun kullanıldığı durumlarda,

farklı aşamalar için oluşturulan dönüşüm fonksiyonları özet olarak aşağıda görülmektedir.

Birinci aşama için

$f_1(s_1) = \min (\max) [r_1, (s_1, x_1)]$

İkinci aşama için

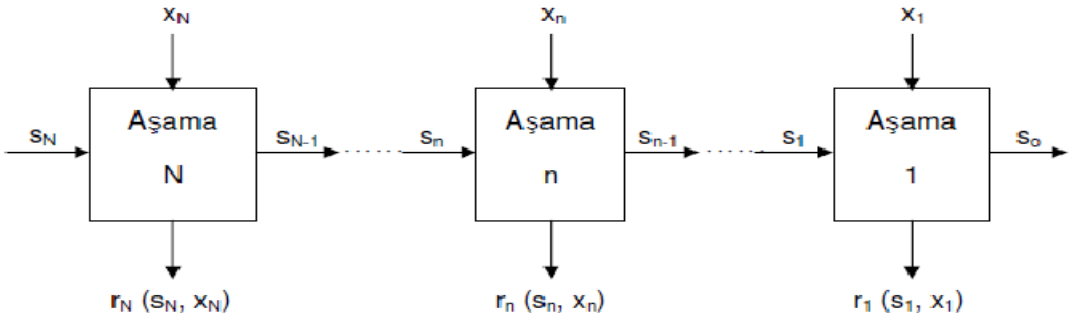
$f_2(s_2) = \min (\max) [r_2 (s_2, x_2)] + f_1 (s_1)$

N inci aşama için

$f_n(x_n) = \min (\max) [r_n (s_n, x_n)] + f_{n-1} (s_{n-1})$

2. Geriye doğru çözüm yöntemi

Bu çözüm yönteminde de çözüme 1 inci aşamadan başlanılarak 2, 3,…, n-1, n ve N inci aşama için işlemler gerçekleştirilir, ancak işlem sırası akısın aksi yöndedir. Geriye doğru çözüm yolu da ileriye doğru çözüm yoluna benzer şekilde şematik olarak aşağıdaki biçimde gösterilebilir [15].



Şekil 2.14. N Aşamalı Bir Problemin Geriye Doğru Çözüm Süreci

Çözüme başlanacak son aşamada sistemin durumu s_1 olarak gösterilirken, bu aşamada gerçekleştirilen dönüşüm fonksiyonu yardımıyla ilk aşamanın çıktısı yani karar değişkeni olan x_1 elde edilir. x_1 kararının verilmesiyle elde edilecek olan getiri de $r_1 (s_1, x_1)$ biçiminde gösterilmektedir. Süreç bu şekilde ilk aşamadan son aşamaya kadar sürdürülür.

Dinamik programlama problemlerinin çözümünde ister tablosal isterse analitik çözüm şekli kullanılsın, aşamalar ileriye veya geriye doğru çözümlenebilir. Her iki yöntem de aynı sonucu vermektedir. Çözümde baştan sona doğru veya sondan başa doğru gidileceğini problemin yapısı ve problemi çözen kişinin probleme yaklaşımı belirlemektedir.

3. Tablosal Çözüm Yolu

Tablosal çözüm yolu optimal politikanın belirlenmesi amacına yönelik olarak, çözümün aşama aşama tablolar halinde gösterilmesi olarak tanımlanabilir. Karar probleminin çözüm süreci ile ilgili aşamalarda tüm durumlar göz önüne alınarak karar seçenekleri belirlenir. Her aşamada dönüşüm fonksiyonları yardımı ile hesaplanan ilgili seçenekler arasından en iyileri belirlenerek tabloya yerleştirilir. Seçenekler arasından seçim yapılırken, seçilen seçeneğin uygun çözüm sağlayıp sağlamadığı da çözüm sırasında göz önünde bulundurulmalıdır. Uygun çözüm sağlamayan seçenekler hesaplamalara dahil edilmez. Dolayısıyla, bu çözüm yolu ile elde edilen çözümler de uygun çözümler olmaktadırlar. Bu nedenle tablosal çözüm yolunun, yalnızca uygun seçenekleri göz önüne alarak çözüm yapılmasına olanak sağladığı söylenebilir.

Tablosal çözüm yolunda her bir tablodaki satırlar, uygun durum değerlerini gösterirken sütunlar da mümkün karar seçeneklerini gösterecek biçimde düzenlenir. Tüm dinamik programlama problemlerinde her aşama için aşağıdakine benzer bir tablo elde edilir. Çözümün ileriye veya geriye doğru en iyileme olmasına göre bir önceki aşamadaki tablo verileri bir sonraki aşamanın tablo çözümünde de kullanılacaktır. Son asamaya ilişkin tablo çözümlendiğinde problemin tamamının çözümü de elde edilmiş olacaktır. Tüm dinamik programlama problemlerinde her aşama için aşağıdakine benzer bir tablo elde edilir.

Tablo 2.16. Tablosal Çözüm

s_n x_n	$r_n(s_n, x_n)$	$f^*_n(s_n)$	$x^*_n(s_n)$

Analitik çözüm yolu

Analitik çözüm yolu, problem için her aşamada oluşturulan dönüşüm fonksiyonlarının türevleri alınarak, her aşamada çözüm için en iyi değerin bulunmaya çalışıldığı çözüm yoludur. Dönüşüm fonksiyonu her aşamada tek bir değişkene bağlı olarak en iyilenmeye çalışılmaktadır. Dinamik programlama ile bir problemin çözümünde tablosal çözümün kullanılabilmesi için problemle ilgili parametrelerin sayısal değerlerinin ve kısıtların açık olarak verilmesi gerekmektedir. Bunların verilmediği durumlarda çözüm yolu olarak analitik çözüm yolu kullanılmaktadır.

Dinamik programlama türleri

1. Deterministik dinamik programlama

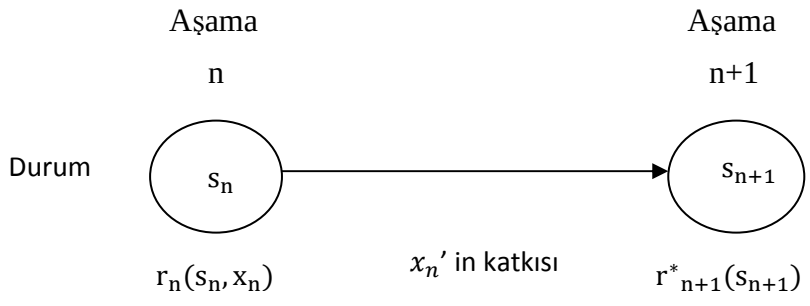
Mevcut aşamada mevcut politika ve durumlarla, gelecek durum ve kararların kesin olarak belirlenebildiği süreçler deterministik süreçler olarak adlandırılmaktadır. Herhangi bir karar probleminde, karar modeli kesin (belirli) bilgilere dayanılarak kuruluyorsa karar süreci deterministik olarak tanımlanır. Bir başka ifadeyle deterministik bir karar modelinde modeldeki katsayılar, kısıtlar ve çözüm değerleri önceden kesin bir şekilde bilinebilmektedir. Deterministik süreçler, bir kararın seçme konusu yapılması durumunda bu karardan doğacak sonuçların önceden bilineceği gerçeğine dayanmaktadır. Doğasının gereği olarak biri ötekisini izleyen süreçler için herhangi bir aşamada durumun değişmesi önceki aşamadaki aksiyon ve durum ile saptanmaktadır. Bunun sonucu olarak, birbirini izleyen deterministik süreçler için kararlar ya bireysel olarak ya da tümüne göre alınabilmektedir. Hiç kuşku yok ki dinamik programlamanın temel özelliğini kararların tek tek ele alınması

oluşturmaktadır. Ancak birbirini izleyen süreçler içerisinde birbirini izlemeyen süreçlerin varlığı da böyle zıt durumu olanaklı kılabilmektedir.

Dinamik programlama yaklaşımında, bir sonraki aşamadaki durum tamamen irdelenmekte olan aşamadaki eylem kararı tarafından belirlenirse problem deterministik olarak tanımlanır. Çok aşamalı karar süreçlerinin çoğunda her karar gider ya da yararlarla ilişkilidir. Söz konusu gider ya da yararlar, sürecin asama ve durumuna göre değişebilmektedir. Bu tür süreçlerin analizindeki temel amaç en büyük toplam getiriyi sağlayan optimal politikanın belirlenmesidir. Deterministik karar sürecinde, her işlemin sonucu kesin olarak bilinmektedir. Her bir alt karar aşamasının optimizasyonu, yöntemine uygun olarak geniş bir alanda kullanılmaktadır. Bir dizi kararların alındığı problemler zaman içinde açıkça belirlenmektedir.

Deterministik programlama problemlerinin içinde üretim planlaması, stok kontrol, yenileme ve yatırım kararlarının alınması problemleri incelenmektedir. Burada dinamik programlama yaklaşımı içindeki bir asama zaman birimi olarak hafta, ay veya yıl ile belirtilebilmektedir. Diğer bir yaklaşıma göre, direkt olarak zamanla belirtmek yerine karar sürecinin sırası önemli olabilmektedir. İkinci yaklaşımın içinde, üretim süreci ve optimal yol problemi sayılabilmektedir.

Dinamik programlama problemleri içinde deterministik problemler oldukça önemlidir. Bu tür problemlerde gelecek aşamada verilecek karar, bugün verilen optimal karar ve durum ile tam olarak belirlenebilmektedir. Deterministik dinamik programlama Şekil 2.15.'de görülmektedir.



Şekil 2.15. Deterministik Dinamik Programlama İçin Temel Yapı

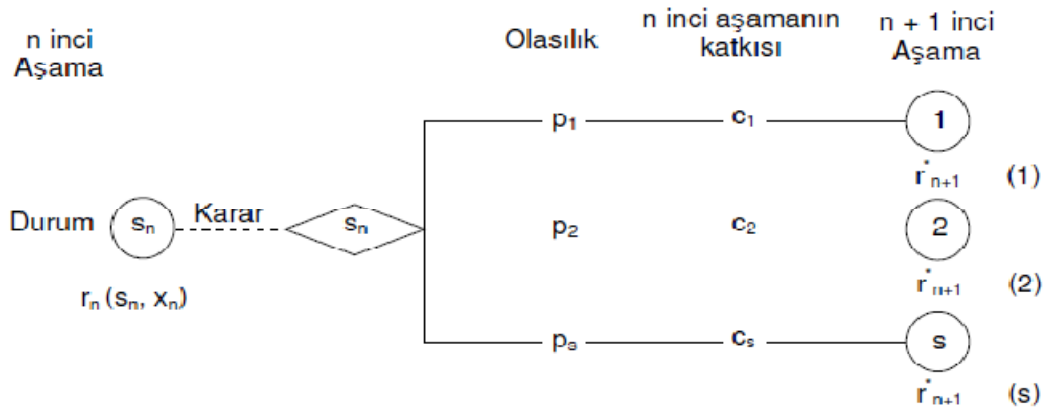
Şekilden de görüleceği gibi n aşamasındaki işlem sn durumunda oluşmaktadır. x_n kararının verilmesi, n aşamasının ileriye doğru amaç fonksiyonuna yaptığı katkı olan r_n (s_n, x_n)' i verir. Bu x_n kararının sürece olan katkısıdır. x_n kararının verilmesi süreci (n+1) aşamasındaki s_{n+1} durumuna götürecektir. Böylece en iyi kararı vermek için oluşan katkı fonksiyonu bir önceki durumdan hesaplanan $r^*_{n+1}(s_{n+1})$olacaktır. x_n için en iyi değeri $r^*_n(s_n) = r_n$ (s_n, r^*_n)' i verir. Sonra s_n'in mümkün her değeri için x^*_n ve $r^*_n(s_n)$ değerleri bulunur. Böylece çözüm, prosedürü bir aşama ileriye götürmeye hazır olmuş olur. Belirli dinamik programlama problemlerini sınıflandırmanın bir yolu amaç fonksiyonunun biçimine göre sınıflandırmadır. Örneğin amaç her bir dönemde toplamları en küçükleme (maliyetler için) veya en büyükleme (kâr için) olabilir. Diğer bir sınıflama ise durum kümelerinin özelliğine göre sınıflandırmadır. s_n durum değişkenleri sürekli, kesikli veya durum vektörü (birden fazla değişken) biçiminde oluşabilir.

1. Stokastik dinamik programlama

İçinde pek çok büyüklükleri bulunduran bir sistemi oluşturan olaylar, zaman içinde tahmin edilemeyen biçimde değişmekte ve olayı oluşturan büyüklüklerin olasılık içinde bulunan değerlerine, hiçbir statik dağılım kanununa bağlı bulunmamaktadır. Bir ya da birden çok büyüklüğün zaman fonksiyonuna göre tesadüfi bir özellik nedeniyle değişmesi durumunda, elde edilen modele "stokastik süreç" adı verilmektedir. Stokastik süreçleri deterministik süreçlerden ayıran en belirgin özellik, bu süreçlerden herhangi bir aşamada verilen karardan ötürü değişen ve ortaya çıkacak durumun daha önceden saptanamamasıdır. Ancak değişen durum belki de yapılan harekete ve ilk duruma bağlı olarak ortaya konan bir olasılık fonksiyonu yolu ile belirginleştirilebilmektedir. Stokastik dinamik programlamada her aşamadaki durumlar ve getiriler olasılıklıdır [7].

Stokastik dinamik programlamanın deterministik dinamik programlamadan en temel farkı, bir sonraki aşamadaki durumun mevcut aşamadaki durum ve politika kararı ile tam olarak belirlenememesidir. Bir sonraki durumda ne olacağına ilişkin bir olasılık dağılımı mevcuttur. Bununla beraber, bu olasılık dağılımı mevcut aşamadaki durum

ve politika karar ile tam olarak belirlenebilmektedir. Şekil 2.16. olasılıklı dinamik optimizasyona ilişkin yapıyı tanımlamaktadır.



Şekil 2.16. Stokastik Dinamik Programlama İçin Temel Yapı

Şekilde s, n+1 aşamadaki mümkün olan durumların sayısı olarak gösterilmiştir ve bu durumlar sağ tarafta 1, 2, … , s ile işaretlenmiştir. Sistem s_n durumu ve n aşamasındaki x_n kararı verildiğinde p_i (i=1, 2, ……., s) olasılığıyla i durumuna gider. Eğer sistem i durumuna giderse, c_i, amaç fonksiyonu için n aşamasında ki katkıdır. Yukarıdaki şekil tüm durum ve kararları içerecek şekilde genişletilirse, şekle bazen karar ağacı da denir. Eğer karar ağacı çok büyük değilse, çeşitli oluşabilecek olasılıkları özetlemek için faydalı bir yol sağlanmış olur.

Olasılıklı yapıdan dolayı, $r_n(s_n, x_n)$ ile $r^*_{n+1}(s_{n+1})$ arasındaki bağıntı deterministik dinamik programlamadakinden daha karmaşıktır. Bağıntının en açık şekli tüm amaç fonksiyonuna dayalı olacaktır. Amacın beklenilen toplamların en küçükleme olduğu varsayılırsa. Bu durumda $r_n(s_n, x_n)$ durumu, s_n durumu ve x_n kararı verildiğinde n aşamasındaki beklenilen değeri verecektir. Buna ilişkin eşitliklerde aşağıdaki gibi olacaktır.

$$r_n(s_n, x_n) = \sum_{i=1}^{x} p_i[\, c_i + r^*_{n+1}(i)]$$

$$r^*_{n+1}(i) = \min r_{n+1}(i, x_{n+1})$$

Burada en küçükleme x_{n+1} değerlerinden oluşacaktır. Stokastik süreçlerde, $x_1, x_2, \ldots, x_n$, rassal değişkenlerde ise, n aşamalı bir süreçte durumsal olasılık; $p_s(x_n = s_n \mid x_{n+1} = s_{n+1})$ n aşamadaki s_n durumu verildiğinde s_{n+1} durumuna hareket edebilme olasılığıdır.

Kabul edilmemiş durumlar için optimal hareketlerin hesaplanması gerekebileceğinden, kimi bilim adamları dinamik programlamanın deterministik sistemler için kullanılmasının gereğinden çok hesaplamalara yol açabileceğini ileri sürmektedir. Buna karşılık stokastik sistem içinde politikanın saptanmasından sonra da olsa hangi durumun kabul edileceği daha önceden bilinememekte; kimi problemlerde, herhangi bir aşamada kabul edilen bir durumun sıfırdan farklı bir olasılığı olabilmektedir. Hiç kuşkusuz böyle durumlarda olanaklı her durum için uygun görülen işlemi bilme gereksinimi vardır. Bu tür sistemler için politikayı saptamada iki olanaklı alternatif yaklaşım vardır. Bunlar ya türlü politikaları benzetim (simülasyon) yolu ile ya da türlü politikaların etkilerini analitik olarak hesaplama yolu ile saptamaktır [13].

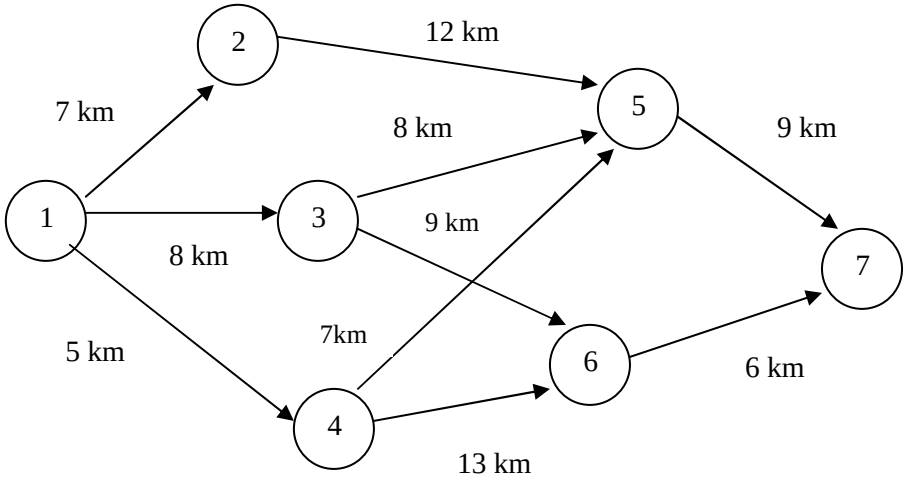
Son yaklaşım Yöneylem Araştırması biliminde “Stokastik Programlama” adı ile anılmaktadır ve stokastik optimumlaştırma problemini özdeş deterministik optimumlaştırma problemi haline dönüştürmektedir. Araştırma konusu yapılan sürecin deterministik bir özelliği olmaması halinde, kararın sonucu durum vektörünün olasılık dağılımına bağlı olarak bulunmaktadır. Böylece, verilen değerler arasında olasılık içinde bir seçim yapma olanağı ortaya çıkmaktadır. Stokastik dinamik programlamada mevcut aşamadaki değer ile bir önceki aşamadaki toplam değer birlikte hesaba katılmakta ve mevcut durumun maliyetleri ile gelecekteki aşamanın durumlarının rassal olduğu varsayımı optimallik prensibine uygulanmaktadır.

Deterministik dinamik programlama ile en kısa yolun bulunması

Dinamik programlama, n değişkenli bir problemin optimum çözümünü, problemi n aşamaya ayırarak ve her aşamada tek değişkenli bir alt problemi çözerek belirler. Hesaplamalar yinelenerek yapıldığı için bir alt problemin optimum çözümü bir

sonraki problemin girdisini oluşturur. Son alt problem çözüldüğünde optimum çözüme ulaşılır. Algoritmada önemli olan konu problemin nasıl parçalara ayrıştırılacağıdır. Bir örnekle algoritma gösterilsin.

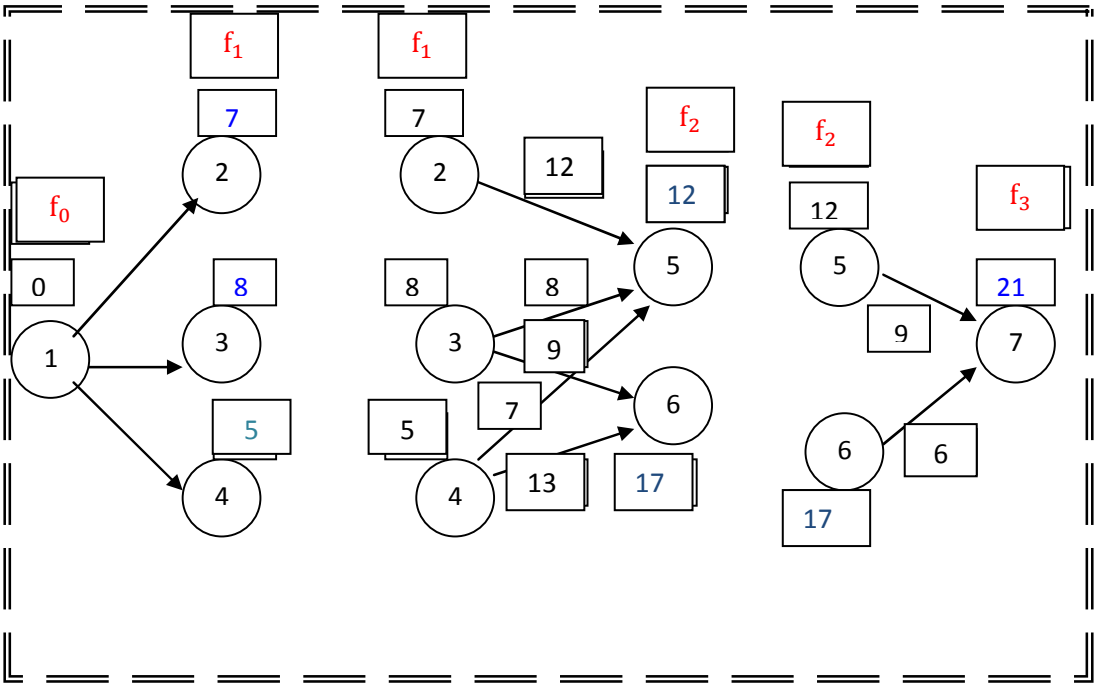
Örnek: Şekil 2.18 de verilen ağ, düğümler ve aralarındaki tüm kullanılabilecek yolları uzaklıkları ile göstermektedir. Birinci düğümden yedinci düğüme kadar giden en kısa yol bulunmaya çalışılacaktır.



Şekil 2.17. Şehirler ve Uzaklıklar

Çözüm

Problem 1. ile 7.düğümler arasındaki tüm yolları tek tek hesaplayarak çözülebilir. Ancak çok büyük bir ağda bu imkansızlaşır. Bundan dolayı problemi matematik olarak ifade edilmelidir. Problemi dinamik programlama ile çözmek için önce parçalara ayırmak gerekir. Yöntemdeki genel düşünce, bir aşamanın (parçanın) son düğümlerine olan en kısa (kümülatif) uzaklıkları hesaplamak, sonra bu değerleri, izleyen aşamada girdi olarak kullanmaktır. Bu problem aşağıda şekil 2.18 de gösterildiği gibi üç aşamaya ayrılabilir.



Şekil 2.18. Aşamalar

1.Aşama:

2.düğüme en kısa uzaklık = 7 km (1.düğümden)

3.düğüme en kısa uzaklık = 8 km (1.düğümden)

4.düğüme en kısa uzaklık = 5 km (1.düğümden)

2.Aşama:

5 ve 6. düğümlere en kısa toplam uzaklıkları belirliyoruz. 5. düğümü ele alırsak üç rota bulunmaktadır. (2,5), (3,5) ve (4,5) bunlardan en kısası şu şekilde belirlenir.

5. düğüme olan en kısa uzaklık = min (i. düğüme olan en kısa uzaklık, i den 5′e olan uzaklık)

i = 2,3,4

5.düğüme olan en kısa uzaklık

2-5 yolu: 12 + 7 = 19
3-5 yolu: 8 + 8 = 16
4-5 yolu: 5 + 7 = 12

6.düğüme olan en kısa uzaklık

3-6 yolu: 8 + 9 = 17
4-6 yolu: 5 + 13 = 18

3.Aşama: varış düğümüne olan (7.düğüm) tüm yolların toplamı bulunmaktadır.

5-7 yolu: 12 + 9 = 21
6-7 yolu: 17 + 6 = 23

7. düğüme en kısa uzaklık 21 km dir. Optimum yolu veren şehirler şu şekilde belirlenir. 3. aşamanın sonucundan hareket ederek 7.düğüm, 5.düğüme bağlanır. Daha sonra 2.aşamanın sonucuna bakılarak 5.düğüm, 4.düğüme bağlanır. Son olarak 1.aşamanın sonucuna bakılarak 4.düğüm, 1. düğüme bağlanır.
Optimum yol $1 – 4 – 5 – 7$ yoludur.

Ele alınan örnekteki hesaplamalar birinci aşamadan başlayıp üçüncü aşamaya geçilerek ileriye doğru bir yineleme ile yapılmıştır. Aynı örnek üçüncü aşamadan başlayıp birinci aşamada bitecek şekilde geriye doğru yineleme ile de çözülebilir. Her iki hesaplama da aynı sonucu verecektir. İleriye doğru hesaplama daha mantıklı görünse de dinamik programlama literatüründe daha çok geriye doğru hesaplamanın kullanıldığı görülmektedir. Bunun nedeni de geriye doğru yinelemenin hesaplama açısından daha etkili olmasıdır. Ele alınan örnek için geriye doğru yineleme denklemi şu şekilde yazılabilir.

i = 1,2,3 olmak üzere tüm uygun (x_i, x_{i+1}) yolları

$f_i(x_i)$ = min($d(x_i, x_{i+1})$ + $f_{i+1}(x_{i+1})$) şeklinde bulunur. Hesaplamalara ait sıra f_3, f_2, f_1 şeklindedir. Aşağıdaki çözüm tablolarında geriye doğru yineleme sırasıyla gösterilmiştir.

Tablo 2.19. 3. Aşama

	$d(x_3,x_4)$	optimum	Çözüm
x_3	$x_4 = 7$	$f_3(x_3)$	x_4
5	9	9	7
6	6	6	7

Tablo 2.20. 2. Aşama

	$d(x_3,x_4) + f_3(x_3)$		optimum	çözüm
x_2	$x_3 = 5$	$x_3 = 6$	$f_2(x_2)$	x_3
2	12+9=21	-	21	5
3	8+9=17	9+6=15	15	6
4	7+9=16	13+6=19	16	5

Tablo 2.21. 1. Aşama

	$d(x_1,x_2) + f_2(x_2)$			Optimum	Çözüm
x_1	$x_2 = 5$	$x_2 = 6$	$x_2 = 4$	$f_1(x_1)$	x_2
1	7+21=28	8+15=23	5+16=21	21	4

Birinci aşamada, birinci şehir dördüncü şehre bağlanıyor. İkinci aşamada optimum çözüm, dördüncü şehrin beşinci şehre bağlandığını gösteriyor. Üçüncü aşamada da beşinci şehir yedinci şehre bağlanıyor. Optimum çözüm (1 - 4 – 5 – 7) numaralı düğümlerden geçmekte ve uzunluğu da 21 km. olmaktadır.

Olasılıklı dinamik programlama örnek problemi

Örnek: Belirsizlik Ortamlarında Optimal Değiştirme Zamanı Stratejisi [18]

C_1: Arizi maliyet tutarı

C_2: Planlama maliyet tutarı

P_k: Beklenen arızalanma olasılığı

F_k: Kümülatif arızalanma olasılığı

R_k: Kümülatif arızalanmama olasılığı

G_t: Ortalama değiştirme maliyeti olmak üzere;

$$G_t = \frac{C_1 - (C_1 - C_2).R_k}{\sum_{i=1}^{k} k.P_k + k.R_k}$$ yoluyla elde edilir.

Tablo 2.22. Probleme Ait Veriler

Dönem	P_k	F_k	R_k	$k.P_k$	$\sum_{i=1}^{k} k.P_k$	$k.R_k$	$\sum_{i=1}^{k} k.P_k + k.R_k$	$C_1 - (C_1 - C_2).R_k$
1	0,1	0,1	0,9	0,1	0,1	0,9	1,0	120
2	0,1	0,2	0,8	0,2	0,3	1,6	1,9	140
3	0,2	0,4	0,6	0,6	0,9	1,8	2,7	180
4	0,1	0,5	0,5	0,4	1,3	2,0	3,3	200
5	0,3	0,8	0,2	1,5	2,8	4,0	6,8	260
6	0,2	1,0	0,0	1,2	4,0	0,0	4,0	300

Tablo 2.23. Probleme Ait Sonuçlar

t (dönem)	**G_t (TL)**
1	120
2	73,68
3	66,66
4	60,60
5	68,42
6	75

Tablo 2.20 de görüldüğü gibi ortalama maliyet 4. Dönem de en az çıktığından optimal değiştirme 4.dönemde yapılmalıdır.

Örnek: 20 dokuma makinesi olan bir işletmenin arızalanan makine olasılıkları poisson dağılımına göre değişmektedir. Günlük ortalama arızalı makine sayısı 3 tür. Bir arıza maliyeti 200 TL/gün ve arıza gün içinde giderilememektedir. Yedek makine maliyeti 50 TL/gün olduğuna göre işletmenin optimal stratejisi nasıl olmalıdır [18].

Çözüm

$$F(x) = \frac{e^{-\lambda}.\lambda^{x}}{x!}$$

$\lambda = 3$

Tablo 2.24. Olasılık Durumları

Arıza Sayısı	0	1	2	3	4	5	6	7	8	9
Olasılık	0,05	0,15	0,224	0,225	0,168	0,114	0,052	0,0216	0,008	0,0027

Tablo 2.25. Yedek Makine Sayıları

A.M.S / Y.M.S	0	1	2	3	4	5	6	7	8	9	Beklenen kapasite eksikliği
0	-	1	2	3	4	5	6	7	8	9	2,981
1		-	1	2	3	4	5	6	7	8	1,991
2			-	1	2	3	4	5	6	7	1,198
3				-	1	2	3	4	5	6	0,660
4					-	1	2	3	4	5	0,310
5						-	1	2	3	4	0,128
6							-	1	2	3	0,0457
7								-	1	2	0,0134
8									-	1	0,008
9										-	0,000

Y.M.S = 0 ise beklenen kapasite eksikliği

(1-0).0,15+(2-0).0,224+(3-0).0,225+…+(9-0).0,0027 = 2,981

Y.M.S = 1 ise beklenen kapasite eksikliği

(2-1).0,224+(3-1).0,225+(4-1).0,168+…+(9-1).0,0027 = 1,991

Y.M.S = 2 ise beklenen kapasite eksikliği

(3-2).0,225+(4-2).0,168+(5-2).0,1+…+(9-2).0,0027 = 1,198

Y.M.S = 3 ise beklenen kapasite eksikliği

(4-3).0,168+(5-3).0,1+(6-3).0,05+…+(9-0).0,0027 = 0,660

Y.M.S = 4 ise beklenen kapasite eksikliği

(5-4).0,1+(6-4).0,05+(7-4).0,0216+…+(9-4).0,0027 = 0,310

Y.M.S = 5 ise beklenen kapasite eksikliği

(6-5).0,05+(7-5).0,0216+(8-5).0,008+(9-5).0,0027 = 0,128

Y.M.S = 6 ise beklenen kapasite eksikliği

(7-6).0,0216+(8-6).0,008+(9-6).0,0027 = 0,0457

Y.M.S = 7 ise beklenen kapasite eksikliği

(8-7).0,008+(9-7).0,0027 = 0,0134

Y.M.S = 8 ise beklenen kapasite eksikliği

(9-8).0,008 = 0,008

Tablo 2.26. Sonuçlar

Y.M.S	**Beklenen Arıza Duruş Maliyeti (TL)**	**Yedek Makine Maliyeti (TL)**	**Toplam Maliyet (TL)**
0	596,2	-	596,2
1	398,2	50	448,2
2	239,6	100	339,6
3	132	150	282
4	62	200	262
5	25,6	250	275,6
6	9,14	300	319,14
7	2,68	350	352,68
8	1,6	400	401,6
9	0	450	450

Yedek tutulması gereken optimal makine sayısı 4 olarak bulunur.

- **Stokastik optimizasyon**

Ortaya konan modellerin gerçek yaşamla uyum sağlayabilmesi belirsizliğin etkin şekilde modellenmesine bağlıdır. Şayet bu belirsiz parametreler kesikli rastgele değişkenler ile tanımlanabilirse stokastik programlama uygulanabilir. Bu tip modellerin çözümünde en etkin yöntemlerden biridir. Stokastik optimizasyon deterministik matematiksel programlamanın doğal ve güçlü bir uzantısıdır ve problem parametrelerinin kesin olarak bilinmediği optimizasyon problemlerinin analizinde etkin olarak kullanılmaktadır. Stokastik programlamada belirsizlik rassal değişkenlerin kesikli dağılımlarla temsil edilir ve rassal değişkenler kesikli belirli bir sayıdaki setten değerler alırlar. Doğal olarak stokastik programlama modellerinin boyutları rassal vektörünün boyutuna göre artar. Bu rasgele vektör ne kadar büyük olursa problemin optimizasyonu da o kadar zorlaşır. Diğer yandan stokastik programlamada çok sayıdaki ihtimallerden en iyisinin tespit edilmesi gerekmektedir ve sadece etkin bilgisayar programları sayesinde bu husus yapılabilmektedir.

Stokastik programlama (SP) modelleri 1950'li yılların sonlarında başta Dantzig (1955), Beale (1955), Charnes ve Cooper (1959) olmak üzere önermiş ve üzerinde çalışılmıştır. Bu yıllarda dinamik yapıdaki stokastik doğrusal programlama ve olasılıklı kısıtlar içeren basit tipteki stokastik programlama modelleri kullanılmıştır. Bunun da ötesinde, olasılık dağılımı (P) bu modellerde yer almıştır. Daha sonraki yıllarda uygulamacılar, deterministik bakış açısını stokastik olanı ile yer değiştirmişler ve bilinmeyen katsayı veya parametreleri, karar değişkenlerinden bağımsız, belli bir olasılık dağılımına sahip rasgele değişkenler olarak ifade etmişlerdir. Stokastik programlama ile ilgili en büyük gelişmeler Prékopa (1978)'nın ortak olasılık kısıtların etkin şekilde kullanılması sayesinde 1970'li yılların sonlarında sağlanmıştır.

Bu yıllarda su kaynakları problemleri ile ilgili modeller yaygın olarak kullanılmıştır. Diğer önemli bir katkı ise çok aşamalı (periyotlu/safhalı) stokastik programlama ile ilgilidir. Bu sayede Yudin (1974), Dempster (1980) ve Prékopa (1985) tarafından

dinamik veya sıralı gerçek karar problemlerinde daha etkin çözümler ortaya konmuştur. Bu yıllarda çok yönlü yaklaşımlar, modern finansal uygulamalarında köşe taşı olarak kabul edilmiş ve çok aşamalı stokastik programlamayla ilgili modelleme ve yazılım gelişmelerine katkıda bulunmuştur.

Takip eden yıllarda hesaplama yöntemlerinde ve bilgisayar teknolojisinde meydana gelen gelişmelere bağlı olarak büyük boyutlardaki problemler yüksek güvenilirlikte etkin olarak çözülmüştür. Bu gelişmeler sayesinde SP yöntemleri gerçek problemlere uygulanabilmektedir.

Modelleme bakış açısıyla incelendiğinde, stokastik programlamada ilk uygulamalar beslenme ve su kaynakları yönetimi modellerinde gözükmesine rağmen takip eden yıllarda stokastik araç rotalama, stokastik şebeke ve stokastik tesis yerleşim ile ilgili uygulamalar daha çok yer almıştır. Günümüzde, stokastik programlama ile ilgili en yaygın uygulamalar finansal uygulamalardır. Diğer uygulama alanları ise, kaynak planlama ve yerleşim problemleri, enerji üretim ve ulaştırma problemleri, teknolojik süreçlerin planlanması ve optimizasyonu, lojistik problemleri ve iletişim problemleridir.

Stokastik programlamada amaç verilen " zor " kısıtlar altında " muhtemel en iyi " kararı vermektir. Burada x karar değişkenin sonucu olarak kabul edilir ve rasgele olayların (ω) etkisiyle değeri oluşur. ω 'nin değeri karar zamanında belli değildir. Karar verici karar verirken, belli olasılık dağılımına(P) sahip ω 'nin değerlerini kullanır.

Kararın rasgele sonuçları, f_0 (x,ω) fonksiyonu ile ifade edilir. Şayet ω 'nin değerleri sınırlı sayıda $\{\omega^1, \omega^2, \dots, \omega^s\}$ ise, çok-amaçlı programlama yöntemi ile f (-,ω^s) fonksiyonu kullanılarak etkin çözümler elde edilebilir. Bu şekildeki etkin çözümler, f_0 (x, ω^s) fonksiyonunun ağırlıklar toplamı minimize veya maksimize edilerek bulunabilir. Stokastik düzenlemede ağırlıklar ω^s, senaryolarının olasılığı p^s olmak üzere ve aşağıdaki amaç fonksiyonunun çözümüyle sonuç elde edilir.

$$\min(\max) \sum_{s=1}^{S} p^s f_0\,(x, \omega^S)$$

Stokastik programlama modelleri; beklenen (anticipative) ve uyarlanabilir (adaptive) karar değişkenlerinin her ikisini de içerebilir. Beklenen değişkenler rasgele parametrelerin gelecekte alacağı değerlerine bağlı değildir ve burada-ve-şimdi (here-and-now) durumlarında oluşturulur. Uyarlanabilir değişkenler, izle-ve-gör(wait-and-see) kararlarıdır ve rasgele parametrelerin gözlemlenmesinden sonra oluşturulur.

Stokastik programlama modeli beklenen değişkenler ve rasgele parametrelere bağlı kısıtları içerdiği durumlarda, değişkenlerin çözümlenebilirliği, tesadüfi kısıtlar kullanılarak tanımlanır. Çoğu gerçekçi model, hem beklenen hem de uyarlanabilir değişkenleri içerebilir. Telafili/Dinamik (Recourse) modeller bu iki tip değişkenlerin bir arada kullanıldığı modellerdir. Dinamik yapıdaki çok aşamalı (safhalı/periyotlu) stokastik programlamanın her safhasında, bilgiler safhalar ilerledikçe ortaya çıkar ve kararlar elde edilen bu bilgilere uyarlanır.
Stokastik programlama modellerinin gelişiminden elde edilen sonuçlar

Stokastik programlama modelleri standart hale getirilmesi (örneğin iki ve çok-aşamalı dinamik stokastik programlama vb.)

Büyük boyuttaki gerçek uygulamaları ortaya konması ve bilgisayar teknolojindeki gelişmeler sayesinde bu tip problemlerin çözülebilmesi,

Bu alanla ilgili geleneksel konferansların yapılması, çalışma gruplarının oluşması, sayısız makalelerin ve araştırmaların yapılması, periyodik bilimsel dergilerin yayınlanmasıdır.

Büyük boyuttaki gerçek uygulamalarının ortaya çıkması yeni sorunları da beraberinde getirmektedir. Burada en önemli zorluk çözümleme sınırlarındaki gelişmeleri de dikkate alan dinamik yapıyı yeterince yansıtmaktır. Diğer bir problem ise, modelin çözülebilir sınırlar içinde kalabilmesi için olasılık dağılımına yakınsayan ve ilgili modeli etkin bir şekilde ifade edebilen yeterli sayıda rasgele değişkenin belirlenmesidir.

- **Kuadratik (karesel) optimizasyon**

Bu optimizasyon probleminde amaç fonksiyonu kuadratik yapıdadır ve model aşağıdaki gösterildiği gibidir.

Zmaks veya Zmin $= CX + X^T DX$

Kısıtlar

$AX \leq b, \qquad X \geq 0$

Burada

$$X = (x_1, x_2, \dots, x_n)^T$$

$$C = (c_1, c_2, \dots, c_n)$$

$$b = (b_1, b_2, \dots, b_m)^T$$

$$A = \begin{bmatrix} a_{11} & \cdots & a_{1n} \\ \vdots & \ddots & \vdots \\ a_{m1} & \cdots & a_{mn} \end{bmatrix}$$

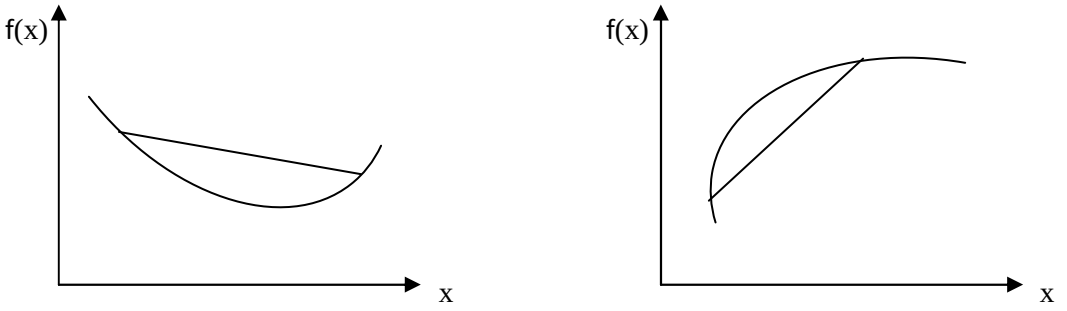
$$D = \begin{bmatrix} d_{11} & \cdots & d_{1n} \\ \vdots & \ddots & \vdots \\ d_{n1} & \cdots & d_{nn} \end{bmatrix}$$

Kuadratik programlama optimizasyonu açısından Kuhn-Tucker koşullarının önemli bir yeri vardır, bu nedenle fonksiyonun konveks ve konkav olma durumu aşağıda kısaca belirtilecektir.

f (x) fonksiyonu, S konveks kümesinde bulunan herhangi x' ve x'' değerleri için,
$0 < \lambda < 1$ aralığında;

$$f(\lambda . x' + (1- \lambda).x'') \leq \lambda.f(x') + (1- \lambda).f(x'')$$

ise f(x) fonksiyona “konveks fonksiyon” adı verilir. Yine bu koşulda $''\leq''$ yerine $'' \geq ''$ geçerliyse, f(x) fonksiyonu konkav fonksiyon olur. Şekil 2.19. de konveks ve konkav fonksiyonlar gösterilmiştir [18].



Konveks Fonksiyon Konkav Fonksiyon

Şekil 2.19. Konveks ve Konkav Fonksiyonlar

Tek değişkenli bir fonksiyonun konkav mı yoksa konveks mi olduğu onun ikinci dereceden türevleri ile belirlenir. Bir fonksiyon farklı bölgelerde konveks veya konkav olabileceğinden belirli bir bölgenin alınması daha anlamlı olmaktadır. Bu nedenle y = f(x) fonksiyonun, bir (a, b) aralığında f"(x) > 0 ise f fonksiyonu bu aralıkta konvekstir, eğer (a, b) aralığında f"(x) < 0 ise f fonksiyonu bu aralıkta konkavdır.

Ör: $f(x) = x^3$, $f'(x) = 3x^2$, $f''(x) = 6x$

$f''(x) = 0 \rightarrow 6x = 0\,,\ x = 0$

x	$-\infty$	0	$+\infty$
f'' (x)	-	0	+

Buna göre (- , 0) aralığında fonksiyon konkav, (0, +∞) da konvekstir. x = 0 noktasında f" (0) = 0, x = 0 ın solunda f "(x) < 0, sağında f"(x) > 0 olduğundan x = 0 noktası büküm noktasıdır.

Kuhn-Tucker koşullarının yeterliliğini sağlayan özellikler aşağıdaki tablo 2.27 de gibi özetlenebilir.

Tablo 2.27. Kuhn Tucker Özellikler

Optimizasyonun yönü	Gereken Koşullar	
	Amaç Fonksiyonu	Çözüm Uzayı
Maksimizasyon	Konkav	Konveks Küme
Minimizasyon	Konveks	Konveks Küme

Bu çerçevede çözüm aşağıdaki matris denklemiyle elde edilir.

$$\begin{bmatrix} -2D & A^T & -I & 0 \\ A & 0 & 0 & I \end{bmatrix} . \begin{bmatrix} X \\ \lambda \\ U \\ S \end{bmatrix} = \begin{bmatrix} C^T \\ b \end{bmatrix}$$

Burada $\lambda = (\lambda_1, \lambda_2, \dots, \lambda_m)^T$ve $U = (\mu_1, \mu_2, \dots, \mu_n)^T$ iki kısıt kümesine karşılık gelen Lagranj çarpanlarıdır.

$X, \lambda, U, S \geq 0$ dır.

Örnek:

$Zmax = 4x_1 + 6x_2 - 2x_1x_2 - 2{x_1}^2 - 2{x_2}^2$

$x_1 + 2x_2 \leq 2$

$x_1, x_2 \geq 0$

Çözüm

Öncelikle başlangıçta verilen Zmaks veya $Zmin = CX + X^TDX$ ve $AX \leq b,\ X \geq 0$ şekliyle problemi matris formuna dönüştürmek gerekir.

$$Zmax = [4 \quad 6] \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + [x_1 \quad x_2] \begin{bmatrix} -2 & -1 \\ -1 & -2 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$$

$X^T DX$ kuadratik bir form tanımlar, burada amaç fonksiyonu maksimizasyon ise D negatif tanımlı, minimizasyon ise pozitif tanımlıdır.

$$[1 \quad 2]\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \leq 2$$

$$x_1, x_2 \geq 0$$

Yukarıda belirtilmiş olan matris denklemine değişkenlerin yerleştirilmesi

$$\begin{bmatrix} -2D & A^T & -I & 0 \\ A & 0 & 0 & I \end{bmatrix} . \begin{bmatrix} X \\ \lambda \\ U \\ S \end{bmatrix} = \begin{bmatrix} C^T \\ b \end{bmatrix}$$

$$\begin{bmatrix} 4 & 2 & 1 & -1 & 0 & 0 \\ 2 & 4 & 2 & 0 & -1 & 0 \\ 1 & 2 & 0 & 0 & 0 & 1 \end{bmatrix} . \begin{bmatrix} x_1 \\ x_2 \\ \lambda_1 \\ \mu_1 \\ \mu_2 \\ S_1 \end{bmatrix} = \begin{bmatrix} 4 \\ 6 \\ 2 \end{bmatrix}$$

Denklemi açılarak iki faz yöntemiyle çözüm gerçekleşir.

Temel	x_1	x_2	λ_1	μ_1	μ_2	R_1	R_2	S_1	Çözüm
r	6	6	3	-1	-1	0	0	0	10
R_1	4	2	1	-1	0	1	0	0	4
R_2	2	4	2	0	-1	0	1	0	6
S_1	1	2	0	0	0	0	0	1	2

Başlangıç tablosu olmak üzere yinelemeler yapıldığında

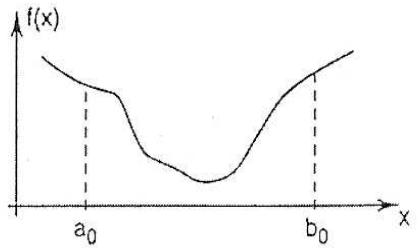
$x_1 = \frac{1}{3}$, $x_2 = \frac{5}{6}$ ve $Z = \frac{25}{6}$ olarak bulunur.

2.1.2. Kısıtsız optimizasyon teknikleri

Doğrusal olmayan optimizasyon yöntemleri aşağıda gösterilmiştir.

- **Golden section (altın oran) metodu**

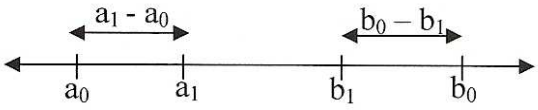
İncelenen bu metod [a_0 , b_0] kapalı aralığında f: R $\rightarrow$R fonksiyonunun minimizerini hesaplanmasını sağlar. Burada tek bir şart olarak fonksiyonun sadece bir lokal minimize noktasına sahip olması gerekir [16].



Şekil 2.20. f(x) Fonksiyonu

İncelenen bu metod [a_0 , b_0] kapalı aralığı içindeki farklı noktalarda amaç fonksiyonunun hesaplanmasını temel alır. Bu noktalar f 'nin minimum noktasına yaklaşacak şekilde seçilmelidir. Amaç aralığı daraltarak minimum noktayı küçük bir aralık içine almaktır.

[a_0, b_0] kapalı aralığında bir değişkenli fonksiyon ele alınsın. Eğer fonksiyon aralıktaki tek bir nokta için hesaplanırsa, minimumu içinde bulunduran bir bölgeye daraltılamaz. Fonksiyon iki ara nokta için hesaplanmalıdır.



Şekil 2.21. İki Ara Nokta

$a_1 - a_0 = b_0 - b_1 = p\,(b_0 - a_0)$ ve $p < \frac{1}{2}$

Biçiminde simetriklik sağlanmalıdır, daha sonra f ara noktalarda hesaplanır.
Eğer ;

$f(a_1) < f(b_1)$ ise minimum nokta $[a_0, b_1]$ aralığında bulunmalıdır. Diğer taraftan

$f(a_1) \geq f(b_1)$ ise minimum nokta $[a_1, b_0]$ aralığında yer almalıdır.

$\frac{a}{a+b} = \frac{b}{a}$ Bu orana altın oran adı verilir

Buradan;

$(a+b)b = a^2$

$ab + b^2 - a^2 = 0$

Biçiminde yazılabilir, a için kökler bulunursa, a'nın pozitif kökü için

$$a = \frac{b(1+\sqrt{5})}{2}$$

$$\tau = \frac{a}{b} = \frac{1+\sqrt{5}}{2} = 1{,}6180833988$$

sayısı elde edilir. Bu sayı altın oran olarak kullanılır. Başlangıç aralığı L_1 olmak üzere, ikinci indirgemede bu aralık $L_2 = \frac{1}{\tau}(b-a)$ uzunluğunda olur, n indirgeme sonucunda ise bu uzunluk $L_n = \frac{1}{\tau^n}(b-a)$ olur. Bu nedenle son belirsizlik aralığının belli bir ε den küçük olması isteniyorsa

$$\frac{1}{\tau^n}(b-a) < \varepsilon$$

yardımıyla araştırma sayısı önceden belirlenebilir.

Bu metod için genel bir algoritma oluşturmak gerekirse;

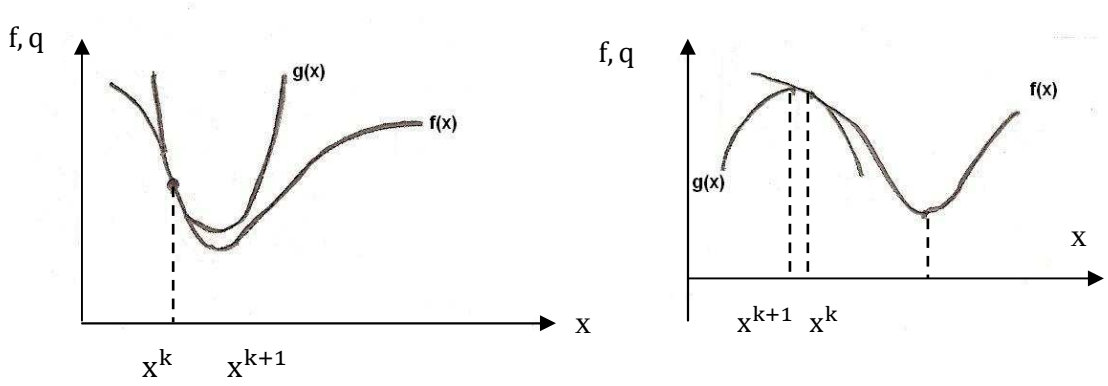
$$a_k = a_1 + (b_0 - a_0)\frac{\tau - 1}{\tau^k}$$

$$b_k = b_1 - (b_0 - a_0)\frac{\tau - 1}{\tau^k}$$

k =1,2,3, ..., N adım sayısı olmak üzere bulunur.

- **Newton metodu**

Bir reel değişkenli fonksiyonun minimize edilmesi problemi ile karşı karşıya olunduğu varsayılsın. Her bir ölçüm noktası $x^{(k)}$ için $f(x^{(k)})$, $f'(x^{(k)})$, $f''(x^{(k)})$ değerleri hesaplanmalı [16].



Şekil 2.22. f(x) ve g(x) Fonksiyonları

Newton metodu her yerde $f''(x) > 0$ olması durumunda daha iyi çalışır. Eğer bazı x ler için $f''(x) < 0$ ise Newton metodu minimum noktaya yaklaşmada hataya düşebilir.

Newton metodu $x^{(k+1)} = x^{(k)} - \frac{f'(x^{(k)})}{f''(x^{(k)})}$ formülüyle uygulanır.

Örnek: $f(x) = \frac{1}{2}x^2 - \sin x$ fonksiyonunun minimum noktasını Newton metodu ile bulunması.

Başlangıç değeri $x^0 = 0{,}5$, $\varepsilon = 10^{-5}$

Çözüm

$\left|x^{k+1} - x^{k}\right| < \varepsilon$ şartı sağlanmalı

$$x^{(k+1)} = x^{(k)} - \frac{f'\left(x^{(k)}\right)}{f''(x^{(k)})}$$

$f'(x) = x - \cos x \qquad f''(x) = 1 + \sin x$

$$x^{(1)} = x^{(0)} - \frac{f'(x^{0})}{f''(x^{0})}, \qquad x^{(1)} = 0{,}5 - \frac{0{,}5 - \cos 0{,}5}{1 + \sin 0{,}5} = 0{,}7552$$

$$x^{(2)} = x^{(1)} - \frac{f'(x^{1})}{f''(x^{1})}, \qquad x^{(2)} = 0{,}7552 - \frac{0{,}02710}{1{,}685} = 0{,}7391$$

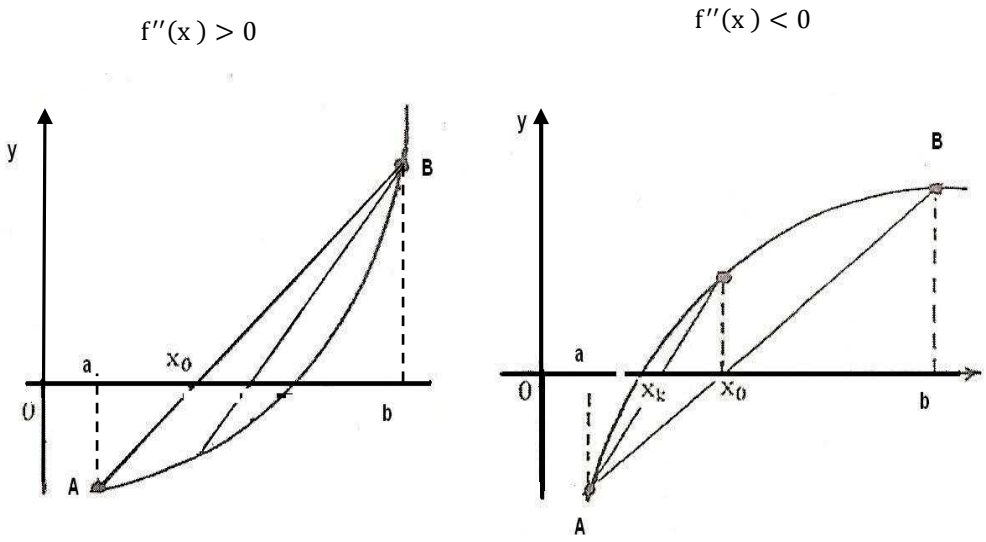
$$x^{(3)} = x^{(2)} - \frac{f'(x^{2})}{f''(x^{2})}, \qquad x^{(3)} = 0{,}7390 - \frac{9{,}461.10^{-5}}{1{,}673} = 0{,}7390$$

$$x^{(4)} = x^{(3)} - \frac{f'(x^{3})}{f''(x^{3})} \qquad x^{(4)} = 0{,}7390 - \frac{1{,}17.10^{-9}}{1{,}673} = 0{,}7390$$

$\left|x^{(4)} - x^{(3)}\right| < \varepsilon$ olduğundan minimum nokta olarak $x^{(4)}$ olarak kabul edilebilir.

- **Sabit kesen (regula falsi) yöntemi**

Değişik işaret yöntemi olarak bilinir. Bu yöntemde gerçek köke yakınsama yavaş olsa da daima yakınsama olduğundan, avantajlı bir yöntem olarak sayılabilir. Yöntem iki tane başlangıç değeri gerektirir. Genellikle ilk iterasyonda aralığın uç noktaları alınıp bu noktalardan geçen kirişin Ox eksenini kestiği nokta yaklaşık kök olarak alınır [16-17].



Şekil 2.23. İkinci türevler

[AB] doğru parçasının eğimi $m = \frac{f(b)-f(a)}{b-a}$ dır.

O halde A [A, f(a)] noktasından geçip eğimi belli olan doğru denklemi

$$y - f(a) = \frac{f(b) - f(a)}{b - a}(x - a)$$

şeklindedir.

Burada $y = 0$ yazılarak doğrunun Ox eksenini kestiği nokta

$$-f(a) = \frac{f(b) - f(a)}{b - a}(x_0 - a)$$

$$x_0 = \frac{a.f(b) - b.f(a)}{f(b) - f(a)} \text{ olarak bulunur}$$

Hesaplanan x_0 noktası daima [a,b] aralığı içinde yer alır.

Örnek: $1 - x - e^{-2x} = 0$ denkleminin [0,5, 1] aralığındaki kökü Sabit Kesen Yöntemiyle araştırılsın.

Çözüm

$f(0{,}5) = 1 - 0{,}5 - e^{-1} = 0{,}133 > 0$

$f(1) = 1 - 1 - e^{-2} = -0{,}135 < 0$ olduğundan

$$x_0 = \frac{0{,}5.f(1) - 1.f(0{,}5)}{f(1) - f(0)} = 0{,}748$$

$f(0{,}748) = 1 - 0{,}748 - e^{-2.(0748)} = -4{,}211 < 0\ dır.$

$f(0{,}5) > 0;$

$f(0{,}748) < 0$

kök [0,5 , 0,748] arasında olup, yeni iterasyon için b değeri 0,748 dir.

- **Gradient arama**

Örnek: $f(x_1, x_2) = 4x_1 + 6x_2 - 2x_1x_2 - 2{x_1}^2 - 2{x_2}^2$ fonksiyonunu (1,1) başlangıç noktasına göre optimum noktasını gradient metodu ile bulun. (2 adım için değişken değerleri bulunacaktır)

Çözüm

$$\nabla f\left(\frac{\delta f}{\delta x_1}, \frac{\delta f}{\delta x_2}\right) = (4 - 2x_2 - 4x_1, 6 - 2x_1 - 4x_2)$$

$x^{(0)} = (1{,}1)$ için

$\nabla f\left(x^{(0)}\right) = (-2,0)$

$x^{(1)} = x^{(0)} + r(-2,0) = (1-2r, 1)$

$f(1-2r, 1) = -8r^2 + 4r + 4$

$f'(x_1, x_2) = -16r + 4 = 0$

$r = \frac{1}{4}$,

$x^{(1)} = (1-2r, 1)\ \ x^{(1)} = \left(\frac{1}{2}, 1\right)$

$\nabla f\left(x^{(1)}\right) = \left(4 - 4.\frac{1}{2} - 2.1\ , 6 - 2.1 - 4.1\right) = (0,1)$

$x^{(2)} = x^{(1)} + r(0,1) = \left(\frac{1}{2}, 1 + r\right)$

$f\left(\frac{1}{2}, 1 + r\right) = -2(1+r)^2 + 5(1+r) + \frac{3}{2}$

$f'(x_1, x_2) = 1 - 4r = 0$

$r = \frac{1}{4}$

$x^{(2)} = x^{(1)} + r(0,1) = \left(\frac{1}{2}, 1 + r\right)$

$x^{(2)} = \left(\frac{1}{2}, \frac{5}{4}\right)$

- **Lagrange çarpanı**

Örnek: $Zmax = 4x_1 - 0{,}1{x_1}^2 + 5x_2 - 0{,}2{x_2}^2$

$x_1 + 2x_2 = 40$

problemi veriliyor. Amaç fonksiyonunu maksimize eden x_1, x_2 değerlerini bulun.

Çözüm

$L(x, \lambda) = 4x_1 - 0{,}1{x_1}^2 + 5x_2 - 0{,}2{x_2}^2 - \lambda(x_1 + 2x_2 - 40)$

$\frac{\delta L}{\delta x_1} = 4 - 0{,}2x_1 - \lambda \qquad 4 - 0{,}2x_1 - \lambda = 0$

$\frac{\delta L}{\delta x_2} = 5 - 0{,}4x_2 - 2\lambda \qquad 5 - 0{,}4x_2 - 2\lambda = 0$

$\frac{\delta L}{\delta \lambda} = -x_1 - 2x_2 + 40 \qquad -x_1 - 2x_2 + 40 = 0$

$x_1 = 18{,}3$

$x_2 = 10{,}8$

$\lambda = 0{,}33$ olarak bulunur.

2.1.2.6. Newton benzeri algoritması (quasi-newton)

Bu algoritma bir fonksiyonun lokal minimum veya maksimum noktasını bulmada kullanılan yaygın bir optimizasyon algoritmasıdır. Optimal eğri uydurma problemi için minimize edilmek istenen fonksiyon S(a) Taylor serisine açılıp yaklaşık değeri hesaplandıktan sonra türevleri alınarak sekant denklemi adı verilen aşağıdaki denklem elde edilir.

$JS(a + q) = JS(a) + Bq$

B: Hessian matrisi

S(a): Amaç fonksiyonu (bu fonksiyonun minimum olduğu noktada türev sıfırdır.

$JS(a + q) = 0$

$q = -B^{-1}JS(a)$

Bu metotta Hessian matrisi (B), birim matris (I) ile başlar ve Hessian matrisinin tersi olan B^{-1} ile biter. Newton-Benzeri algoritmasında Hessian'ın yaklaşık değerini hesaplamak için Symmetric rank 1 formülü (SR1) ve Broyden-Fletcher-Goldfarb-Shanno (BFGS) formülleri kullanılır.

Bu algoritma metodları belirli bir eğim çizgisi boyunca hareket ederek, eğim bilgisini kullanıp çözüm bulmaya çalışır. Bu noktada aşağıdaki denklem önemlidir.

$$a_{k+1} = a_k + q_{k+1} B_k JS(a_k)$$

Örnek: $f(x) = 5{x_1}^2 + 2{x_2}^2 + 2{x_3}^2 + 2x_1x_2 + 2x_2x_3 - 2x_1x_3 - 6x_3$ fonksiyonunu BFGS metodlu Newton-Benzeri algoritmasıyla minimumlaştırın. X başlangıç değerleri (0,0,0) dır.

Çözüm

$$Jf(x) = \begin{bmatrix} 10x_1 & 2x_2 & -2x_3 \\ 2x_1 & 4x_2 & 2x_3 \\ -2x_1 & 2x_2 & 4x_3 - 6 \end{bmatrix}$$

$$B = \begin{bmatrix} 10 & 2 & -2 \\ 2 & 4 & 2 \\ -2 & 2 & 4 \end{bmatrix}$$

$$B = \begin{bmatrix} 10.0 & 2.0 & 2.0 \\ 2.0 & 4.0 & 2.0 \\ -2.0 & 2.0 & 4.0 - 6 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ -6 \end{bmatrix}$$

$$\begin{bmatrix} x_{11} \\ x_{21} \\ x_{31} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} - q_1 \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ -6 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 6q_1 \end{bmatrix}$$

$x_1 = 0, x_2 = 0, x_3 = 6q_1$ değerleri amaç fonksiyonunda yerine koyulup, amaç fonksiyonunun türevi alınıp sıfıra eşitlendiğinde $q_1$elde edilir.

$f(x) = 2.(6q_1)^2 - 6.(6q_1)$

$$\frac{df(x)}{dq_1} = 144q_1 - 36 = 0$$

$$q_1 = \frac{1}{4}$$

q_1 değeri yerine koyulduğunda ${x_1}^T = (0, 0, \frac{3}{2})$ olarak bulunur. Buna göre $Jf(x_1)^T = (-3, 3, 0)$olur. BFGS' ye göre Hessian matrisi aşağıdaki aşağıda ki formülle elde edilir.

$\delta_k = x_{k+1} - x_k$ ve $\gamma_k = Jf(x_{k+1}) - Jf(x_k)$ olarak tanımlanır ve yeni Hessian matrisi

$$B_{k+1} = B_k - \left[\frac{B_k\,\gamma_k\delta_k{}^T + \delta_k\gamma_k{}^T B_k}{\delta_k{}^T\gamma_k}\right] + \left[1 + \frac{\gamma_k{}^T B_k\gamma_k}{\delta_k{}^T\gamma_k}\right]\left[\frac{\delta_k\delta_k{}^T}{\delta_k{}^T\gamma_k}\right]$$

formülüyle elde edilir. Bir sonraki iterasyondaki B_1matrisi;

$$\gamma_k = x_{k+1} - x_k = \begin{bmatrix}-3\\3\\0\end{bmatrix} - \begin{bmatrix}0\\0\\-6\end{bmatrix} = \begin{bmatrix}-3\\3\\6\end{bmatrix}$$

$$\delta_k = Jf(x_{k+1}) - Jf(x_k) = \begin{bmatrix}0\\0\\\frac{3}{2}\end{bmatrix} - \begin{bmatrix}0\\0\\0\end{bmatrix} = \begin{bmatrix}0\\0\\\frac{3}{2}\end{bmatrix}$$

$$B_1 = \begin{bmatrix}1&0&0\\0&1&0\\0&0&1\end{bmatrix} - \left(\frac{\begin{bmatrix}1&0&0\\0&1&0\\0&0&1\end{bmatrix}\begin{bmatrix}-3\\3\\6\end{bmatrix}\begin{bmatrix}0&0&\frac{3}{2}\end{bmatrix} + \begin{bmatrix}0\\0\\\frac{3}{2}\end{bmatrix}\begin{bmatrix}-3&3&6\end{bmatrix}\begin{bmatrix}1&0&0\\0&1&0\\0&0&1\end{bmatrix}}{\begin{bmatrix}0&0&\frac{3}{2}\end{bmatrix}\begin{bmatrix}-3\\3\\6\end{bmatrix}}\right) +$$

$$\left(1 + \frac{\begin{bmatrix}-3&3&6\end{bmatrix}\begin{bmatrix}1&0&0\\0&1&0\\0&0&1\end{bmatrix}\begin{bmatrix}-3\\3\\6\end{bmatrix}}{\begin{bmatrix}0&0&\frac{3}{2}\end{bmatrix}\begin{bmatrix}-3\\3\\6\end{bmatrix}}\right)\left(\frac{\begin{bmatrix}0\\0\\\frac{3}{2}\end{bmatrix}\begin{bmatrix}0&0&\frac{3}{2}\end{bmatrix}}{\begin{bmatrix}0\\0\\\frac{3}{2}\end{bmatrix}\begin{bmatrix}-3\\3\\6\end{bmatrix}}\right)$$

$$B_1 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} - \left(\frac{\begin{bmatrix} 0 & 0 & -\frac{9}{2} \\ 0 & 0 & \frac{9}{2} \\ 0 & 0 & 9 \end{bmatrix}}{9} \right) + 7 \left(\frac{\begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & \frac{9}{4} \end{bmatrix}}{9} \right)$$

$$B_1 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} - \left(\frac{\begin{bmatrix} 0 & 0 & -\frac{9}{2} \\ 0 & 0 & \frac{9}{2} \\ -\frac{9}{2} & \frac{9}{2} & 18 \end{bmatrix}}{9} \right) + 7 \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & \frac{1}{4} \end{bmatrix}$$

$$B_1 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} - \begin{bmatrix} 0 & 0 & -\frac{1}{2} \\ 0 & 0 & \frac{1}{2} \\ -\frac{1}{2} & \frac{1}{2} & 2 \end{bmatrix} + \left(\begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & \frac{7}{4} \end{bmatrix} \right)$$

$$B_1 = \begin{bmatrix} 1 & 0 & \frac{1}{2} \\ 0 & 1 & -\frac{1}{2} \\ -\frac{1}{2} & \frac{1}{2} & \frac{3}{4} \end{bmatrix}$$

olur. Sonraki adımlar bu şekilde devam ettirilir. Problem parametrelerinin sonraki adımlar için aldığı değerler aşağıda gösterilen Tablo 2.28 de ki gibi olur.

Tablo 2.28. Sonraki Adım Değerleri

k	x_1	x_2	x_3	**q**	**f(x)**
0	0	0	0	-	0
1	0	0	$\frac{3}{2}$	$\frac{1}{4}$	-4,5
2	1	-1	$\frac{5}{2}$	$\frac{1}{3}$	-7,5
3	1	-2	3	$\frac{5}{12}$	-9

BÖLÜM 3.

ZEKİ OPTİMİZASYON TEKNİKLERİNİN SINIFLANDIRILMASI VE BU TEKNİKLERLE PROBLEMLERİN ÇÖZÜMÜ

3.1. Giriş

Çağdaş dünyada bilgisayarlar ve bilgisayar sistemleri yaşamın vazgeçilmez bir parçası haline gelmiştir. Günlük hayatta kullanılan elektronik aletler bilgisayar sistemleri ile çalışmaktadır. İş dünyasından kamu işlerine, çevre ve sağlık organizasyonlarından askeri sistemlere kadar hemen hemen her alanda bilgisayarlardan faydalanmak olağan hale gelmiştir. Teknolojinin gelişmesi izlendiğinde önceleri sadece elektronik veri transferi yapmak ve karmaşık hesaplamaları gerçekleştirmek üzere geliştirilen bilgisayarların, zaman içerisinde büyük miktarlardaki verileri filtreleyerek özetleyebilen ve mevcut bilgileri kullanarak olaylar hakkında yorumlar yapabilen nitelikler kazandığı, ayrıca hem olaylar hakkında karar verebilmekte hem de olaylar arasındaki ilişkiyi öğrenebilmekte olduğu görülmektedir. Matematiksel olarak formülasyonu kurulamayan ve çözülmesi mümkün olmayan problemler zeki optimizasyon yöntemleriyle çözülebilmektedir. Bunu sağlayan çalışmalar “ yapay zeka ” çalışmaları olarak bilinmektedir [19]. İlk defa 1954 yılında John McCarthy tarafından ortaya atılan bu çalışmalar zamanla geliştirilerek geniş bir alana yayılmıştır.

Yapay zeka çalışmaları artık bilgisayarların sadece bilgi iletişiminin ve hesaplamaların otomasyonunu yapan sistemler olmaktan öteye götürerek, artan donanım teknolojisiyle birlikte bilgi işleme, öğrenme, karar verebilme, problem çözme ve muhakeme yapabilme yetenekleri katmıştır.

Zeki optimizasyon teknikleri Newell ve Pazer tarafından hepsinin aynı anda bir arada bulunması zorunlu olmayan, üç alt unsurun birleşimi olarak tanımlanır.

Bulgusal problem çözme: Bir problemin tatmin edici bir çözüme ulaşması için problemin yönlendirilmesidir. Yönlendirilmeden kast edilen nokta karar vericinin düşünce sürecini adım adım tekrarlamaktansa, zeki davranış elde etmek üzere bilgisayarların etkin kullanmaya yönelmesi olarak düşünülebilir [19].

Öğrenme: Örüntü tanıma (pattern recognition) gibi çözüm uzayı üzerinde arama gerektiren problemlerde öğrenme ve tümevarımsal sonuç çıkarma önemlidir [19].

İnsan düşüncesine benzetim: Karar vericinin, insan düşünce sürecinin tekrarlaması olarak tanımlanır [19].

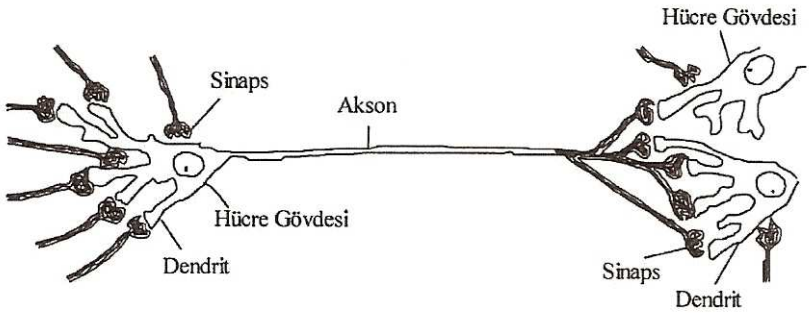
Zeki optimizasyon tekniklerinin en temel özelliği problemler çözüm üretirken veya üretmeye çalışırken bilgiye dayalı olarak karar verebilme özelliklerinin olması ve eldeki bilgiler ile olayları öğrenerek, sonraki olaylar hakkında kararlar verebilmesidir. Bu noktadan hareketle zeki optimizasyon, problemin çözümünü veren bir formülün olmadığı durumlarda problemin çözümünü üstlenme yeteneğine sahiptir [22-24].

Bu bağlamda zeki optimizasyon teknikleri tez çalışmasında aşağıdaki gibi sınıflandırılmıştır.

- Yapay Sinir Ağları (Neural Networks)
- Genetik Algoritmalar (Genetic Algorithms)
- Bulanık Optimizasyon (Fuzzy Logic)
- Sürü Optimizasyonu (Swarm Optimization)
- Tavlama Benzetimi Algoritması (Simulated Annealing)
- Tabu Arama Optimizasyon Algoritması (Tabu Search Algorithm)
- Karınca Koloni Optimizasyonu (Ant Colony Optimization)

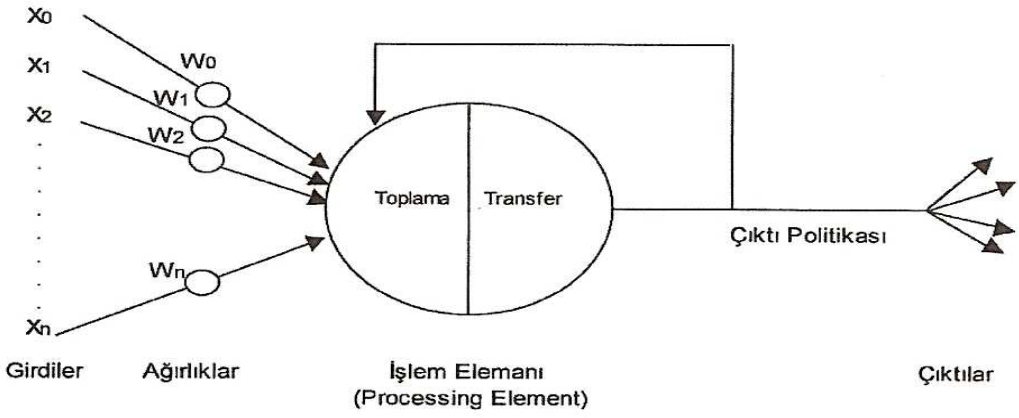
3.2. Yapay Sinir Ağları

Yapay Sinir ağları insan beyninin en temel özelliği olan öğrenme fonksiyonunu gerçekleştiren sistemdir. Bu nedenle biyolojik sinir sisteminden esinlenerek ortaya çıkarılmış bilgi işlem mekanizmasıdır [20].



Şekil 3.1. Biyolojik Sinir Ağı Yapısı

Yukarıda biyolojik sinir ağının yapısı gösterilmiştir. YSA bu ağ yapısı simule edilerek tasarlanmıştır. YSA birbirine bağlı proses elemanlarından (yapay sinir hücreleri) oluşur. Her bağlantının bir ağırlık değeri vardır. YSA' nın sahip olduğu bilgi bu ağırlık değerlerinde saklı olup ağa yayılmıştır. YSA yapısı aşağıdaki gibidir.



Şekil 3.2. Yapay Sinir Ağı Yapısı

Yukarıda ki Şekil 3.2. de görüldüğü gibi bir YSA 5 ana kısımdan oluşur.

Girdiler: Yapay sinir hücresine (proses elemanına) dış dünyadan gelen bilgilerdir. Bunlar ağın öğrenmesi istenen örnekler tarafından belirlenir.

Ağırlıklar: Bir yapay hücreye gelen bilginin önemini ve hücre üzerinde etkisini gösterir. Örneğin yukarıda şekilde w_1 (ağırlık 1); x_1 hücresi (girdi 1) üzerindeki etkisini göstermektedir. Ağırlıkların büyük ya da küçük olması önem ya da önemsizlik arz etmez. Bir ağırlık değerinin sıfır olması, o ağ için en önemli olay olabilir. Ağırlığın eksi veya artı olması, etkinin pozitif veya negatif olduğunu verir.

Toplama fonksiyonu: Bu fonksiyon bir hücreye gelen net girdiyi hesaplar. Bunun için değişik fonksiyonlar kullanılmaktadır. En yaygın olanı ağırlıklı toplamı bulmaktır.

$$\mathrm{NET} = \sum_{i=1}^{n} G_i A_i$$

Bu şekilde formülize edilmektedir. Burada

G: Girdiler
A: Ağırlıklar
n: Bir hücreye gelen toplam girdi sayısını göstermektedir.

Bir YSA da bulunan proses elemanlarının tamamının aynı toplam fonksiyonuna sahip olmaları gerekmez. Her proses elemanı bağımsız olarak farklı bir toplama fonksiyonlarına sahip olacakları gibi hepsi aynı fonksiyona sahip olabilir.

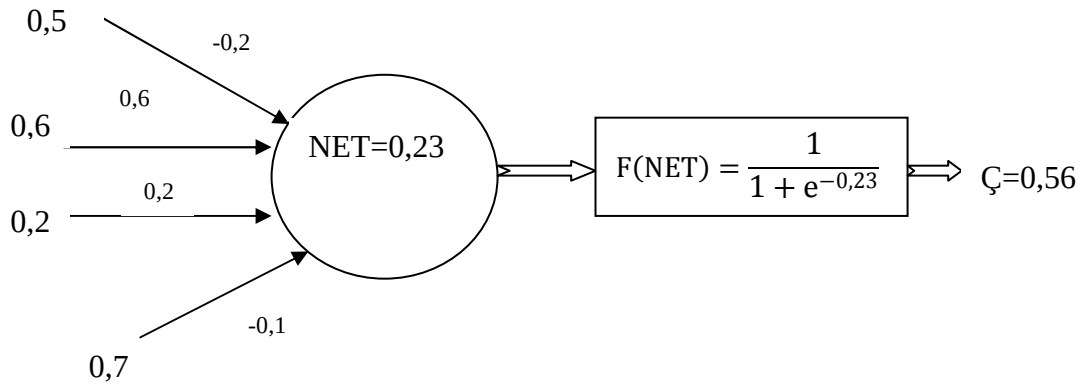
Aktivasyon fonksiyonu (Transfer Fonksiyonu): Bu fonksiyon hücreye gelen net girdiyi işleyerek hücrenin bu girdiye karşılık üreteceği çıktıyı belirler. Toplama fonksiyonunda olduğu gibi aktivasyon fonksiyonunda da ağın proses elemanlarının hepsinin aynı fonksiyonu kullanması gerekmez. En yaygın aktivasyon fonksiyonu olarak sigmoid fonksiyon kullanılmaktadır. Burada ki önemli nokta bir problem için aktivasyon fonksiyonunun tasarımcının denemeleri sonucunda belirleyebileceğidir. Bu fonksiyon aşağıdaki gibidir [22].

$$F(NET) = \frac{1}{1+e^{-net}}$$

Çıktılar: Aktivasyon fonksiyonu tarafından belirlenen çıktı değeridir.

3.2.1. Bir yapay sinir hücresinin çalışma prensibi

Bir YSH nin (proses elemanı) nasıl çalıştığını anlamak için Şekil 3.3 te örnek model gösterilmiştir.

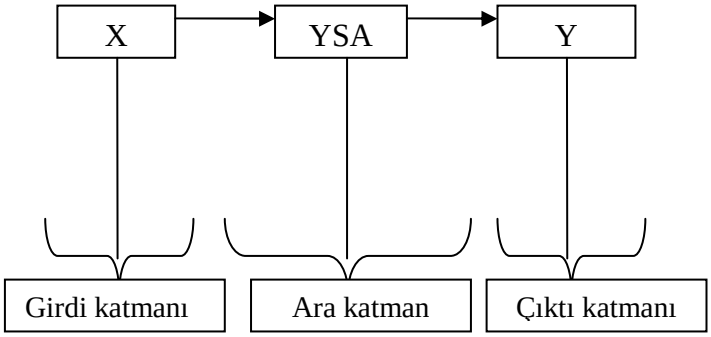


Şekil 3.3. Örnek Model

$$NET = 0{,}5.(-0.2) + 0{,}6.0{,}6 + 0{,}2.0{,}2 + 0{,}7.(-0{,}1) = 0{,}23$$

Bir ağdaki bütün proses elemanlarının çıktılarının bu şekilde hesaplanması sonucu ağın girdilere karşı çıktıları nasıl ürettiği görülür.

Yapay sinir ağlarının genel çalışma prensibi, bir girdi setini (örnekleri) alarak onları çıktı setine çevirmek olarak açıklanır. Bunun için ağın kendisine gösterilen girdiler için doğru çıktıları üretecek hale gelmesi(eğitilmesi) gerekmektedir. Ağa gösterilecek örnekler öncelikle bir vektör haline getirilir. Bu vektör ağa gösterilir ve ağ bu vektör için gerekli çıktı vektörünü üretir. Ağın parametre değerleri doğru çıktıyı üretecek şekilde düzenlenir. Girdi ve çıktı vektörlerinin tasarımı, ağı geliştiren kişi tarafından belirlenir [20].



Şekil 3.4. YSA' da Katmanlar

Girdi vektörü: $X = (X_1, X_2, X_3, \dots, X_n)^T$

Çıktı vektörü: $Y = (Y_1, Y_2, Y_3, \dots, Y_n)^T$

Bilgiler ağa girdi katmanından iletilir. Ara katmanlarda işlenerek oradan çıktı katmanına gönderilir. İşlemeden kast edilen ağa gelen bilgilerin ağın ağırlık değerleri kullanılarak çıktıya dönüştürülmesidir. Ağın girdiler için doğru çıktıları üretebilmesi için ağırlıkların doğru değerlerinin olması gerekmektedir. Doğru ağırlıkların bulunması işlemine ağın eğitilmesi denir. Bu değerler başlangıçta rastgele atanırlar, eğitim sırasında her örnek ağa gösterildiğinde ağın öğrenme kuralına göre ağırlıklar değiştirilir. Daha sonra başka bir örnek ağa sunularak ağırlıklar yine değiştirilir ve en doğru değerler bulunmaya çalışılır. Bu sağlandıktan sonra test setindeki örnekler ağa gösterilir. Eğer ağ test setindeki örneklere doğru cevaplar verirse ağ eğitilmiş kabul edilmektedir.

Mühendislik açıdan bir noktanın altını çizmek gerekmektedir. Ağın ağırlıkları tek tek belirlendikten sonra her bir ağırlığın ne anlama geldiği ve bir YSA da herhangi bir girdi vektörünün çıktı vektörüne nasıl dönüştüğü bilinmez. Bununla beraber ağın girdiler hakkındaki kararını bu ağırlıkları kullanarak vermesi, ağın zekasının bu ağırlıklarda saklı olduğunu gösterir. Ağın bir olayı öğrenmesi o olay için en doğru YSA modelini seçmekle mümkündür.

Bir YSA aşağıda sayılan yapılarla karakterize edilir.

Ağın topolojisi
Kullanılan toplam fonksiyonu
Kullanılan aktivasyon fonksiyonu
Öğrenme stratejisi
Öğrenme kuralı

3.2.2. Öğrenme stratejileri

Genel olarak üç öğrenme stratejisi vardır.

1. Öğretmenli öğrenme (Supervised): Her örnek için hem girdiler hem de o girdiler karşılığında oluşturulması gereken çıktılar sisteme gösterirler. Sistemin görevi girdileri öğretmenin belirlediği çıktılara hatırlatmaktır. “ Çok Katmanlı Algılayıcı ” ağı bu stratejiyi kullanan ağlara örnek olarak verilir.

2. Destekleyici öğrenme (Reinforcement): Her girdi seti için olması gereken (üretilmesi gereken) çıktı setini sisteme göstermek yerine, sistemin kendisine gösterilen girdilere karşılık çıktısını üretmesini bekler ve üretilen çıktının doğru veya yanlış olduğunu gösteren bir sinyal üretir. Bu stratejiyi kullanan ağa örnek olarak “LVQ Ağı ” gösterilir.

3. Öğretmensiz öğrenme (Unsupervised): Örneklerdeki parametreler arasındaki ilişkileri sistemin kendi kendisine öğrenmesi beklenir. Bu daha çok sınıflandırma problemleri için kullanılan bir stratejidir. Örnek olarak “Art Ağları” verilir.

Öğrenme kuralı olarak YSA da genel olarak “ Delta Öğrenme Kuralı “ kullanılır. Bu kurala göre beklenen çıktı ile gerçekleşen çıktı arasındaki farlılığı azaltmak için yapay sinir ağının elemanlarının bağlantılarının ağırlık değerlerinin sürekli değiştirilmesi ilkesine dayanarak geliştirilmiştir. Ağın ürettiği çıktı ile üretilmesi gereken çıktı arasındaki hatanın karelerinin ortalamasın en aza indirmek hedeflenmektedir.

3.2.3. Yapay sinir ağının uygulama alanları

YSA endüstri, finans, askeri ve sağlık alanlarında geniş uygulama alanına sahiptir. Bu alanlarda YSA ile ilgili birçok çalışma yapılmıştır. Bahsedilen alanlarda ki uygulamaların bazıları aşağıdaki gibidir.

Üretim planlama ve çizelgeleme
Zeki araç rotalama ve robotlar için rota belirleme
Pazar verilerinin analizi ve tahmini
Banka kredilerinin değerlendirilmesi
İmza tanıma sistemleri
Veri madenciliği
Hedef tanıma ve takip sistemleri
Transplant zamanlarının optimizasyonu
Hamile bayanların karınlarındaki çocukların kalp atışlarının izlenmesi

3.2.4. Yapay sinir ağı uygulamalarının avantajları

YSA matematik olarak modellenmesi mümkün olmayan veya zor olan karmaşık problemleri modelleyerek çözebilir.

Gerçek dünyada olaylar ve olayların arkasındaki değişik faktörlerin birbirleri ile ilişkilerini ve birbiri üzerindeki etkilerini gerçek hayatta bilmek zordur. YSA bu ilişkileri otomatik olarak örneklerden öğrenir. Kullanıcını bu ilişkiyi bilmesi ve ağa söylenmesi beklenmez.

YSA yeni bilgilerin ortaya çıkması ve ortamda bazı değişikliklerin olması durumunda yeniden eğitilebilirler.

YSA nın paralel çalışabilmeleri onların gerçek zamanlı kullanımını kolaylaştırmaktadır.

3.2.5. Yapay sinir ağı uygulamalarının dezavantajları

Problemlerin YSA ile çözülebilmesi ve örneklerin tasarlanması için bir kural seti yoktur. Problemin sahibi kendi tecrübesine göre örnekleri formülize etmektedir. Doğru bir şekilde gösterimin yolu yine tecrübeler ile sınırlıdır.

Örneklerin bulunmasının güç olduğu durumlarda veya problemi doğru temsil eden örneklerin bulunmaması durumunda problemlere sağlıklı çözümler üretebilmek mümkün olmamaktadır.

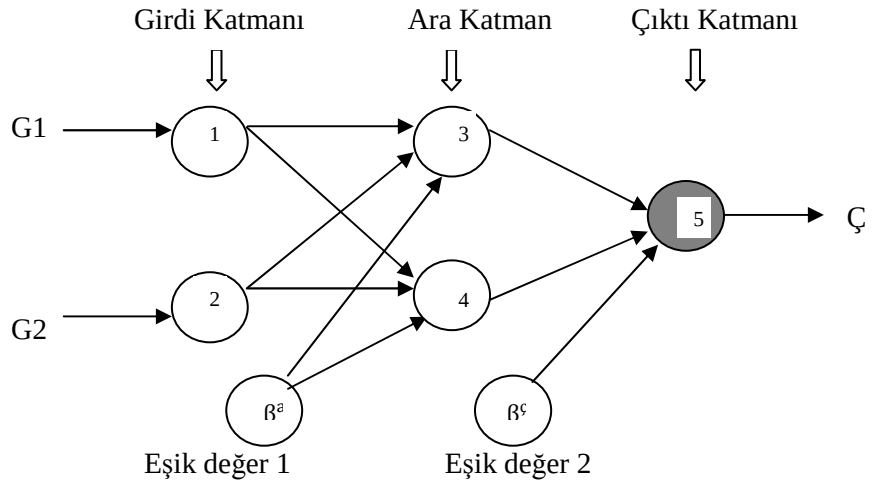
3.2.6. Çok katmanlı yapay sinir ağı örneği

YSA ile ilgili temel bilgiler verildikten sonra, bu ağın temelini meydana getiren problem örneği aşağıdaki gibidir.

Tablo 3.1. XOR gösterimi

	Girdi 1	Girdi 2	Çıktı
Örnek 1	0	0	0
Örnek 2	0	1	1
Örnek 3	1	0	1
Örnek 4	1	1	0

Bu örnek tipinde ağın topolojik yapısının iki girdi ünitesi ve bir çıktı ünitesi, bir ara katman ve iki tanede ara katman proses elemanı ile çözüme ulaşacağı varsayılıyor [20-22].



Şekil 3.5. YSA' da Topolojik Gösterim

Ağ yapısı şekildeki gibi oluşturulduktan sonra ağ için aktivasyon fonksiyonunun sigmoid, parametre değerleri öğrenme (λ) ve momentum (α) katsayıları sırasıyla 0,5 ve 0,8 dir.

Girdi katmanı ile ara katman arasındaki ağırlıklar

$$W^{a} = \begin{bmatrix} 0{,}129952 & 0{,}570345 \\ -0{,}923123 & 0{,}328392 \end{bmatrix}$$

Ara katman ile çıktı katmanı arasındaki ağırlıklar

$$W^{c} = [0{,}164732 \quad 0{,}752621]$$

Eşik değer ağırlıkları

$$\beta^{a} = [0{,}341332 \quad -0{,}115223]$$

$$\beta^{c} = [-0{,}993423]$$

Matrisleriyle belirlendikten sonra örnek 1'in girdi ve çıktı değerleri baz alınarak ileriye doğru hesaplamaya başlanılabilir.

$$NET3 = 0.(-0.129952) + 0.(-0{,}923123) + 0{,}341332 = 0{,}341332$$

$\mathrm{NET4} = 0.(0.570345) + 0.(0.328392) - 0.115223 = -0.115223$

Ara katman ünitelerinin çıktıları ise

$$Ç_3 = \frac{1}{1 + e^{0,341332}} = 0,584490$$

$$Ç_4 = \frac{1}{1 + e^{0,115223}} = 0,471226$$

Çıktı proses elemanının net girdisi hesaplanırsa

$\mathrm{NET\ Ç} = (0,584490).(0,164732) + (0,471226).(0,7526621) - 0,993423 = 0,341332$

$$Ç = \frac{1}{1 + e^{0,341332}} = 0,367610$$

Beklenen çıktı 0 olduğundan ağın hatası;

$E = B - Ç = 0 - 0,367610 = -0,367610$ olur.

Bu noktada hatayı geriye doğru yayarak ağın eğitilmesi sağlanır. Hata miktarı çeşitli yollarla (Gradyan azalan öğrenme ağları, Momentumlu Gradyan azalan öğrenme ağları, Adaptasyon kabiliyetli Gradyan öğrenme ağları, Levenberg-Marquardt öğrenme ağları gibi)

Burada momentumlu azalan öğrenme ağı kullanarak hata şu şekilde geriye yayılmıştır.

$\delta_5 = Ç.(1 - Ç).E_1$
$\delta_5 = 0,367610.(1 - 0,367610).(-0367610)$
$\delta_5 = -0,085459$ (çıktı katmanındaki hata değeri)

Bu hata değeriyle ağırlıklar şu şekilde tekrar hesaplanacaktır.

$$\Delta W_{jm} = \lambda . \delta_m . Ç_J + \alpha . \Delta W_{jm}(t-1)$$

$$\Delta W_{35} = 0{,}5 . \delta_5 . Ç_3 + 0{,}8 . \Delta W_{35}(t-1)$$

$$\Delta W_{35} = 0{,}5 . (-0{,}085459) . 0{,}584490 + 0{,}8.0 = -0{,}024975$$

$$\Delta W_{45} = 0{,}5 . \delta_5 . Ç_4 + 0{,}8 . \Delta W_{45}(t-1)$$

$$\Delta W_{45} = 0{,}5 . (-0{,}085459) . 0{,}471226 + 0{,}8.0 = -0{,}020135$$

$$NW_{35} = W^{c3} + \Delta W_{35}$$

$$NW_{35} = 0{,}164732 - 0{,}24975 = 0{,}139757$$

$$NW_{45} = 0{,}752621 - 0{,}020135 = 0{,}732486$$

$$\Delta \beta^{ç} = \lambda . \delta_ç . + \alpha . \Delta \beta(t-1)$$

$$\Delta \beta^{ç} = 0{,}5 . (-0{,}085459) + 0{,}8.0 = -0{,}0427295$$

$$\beta^{ç} = -0{,}993423 - 0.0427295 = -1{,}036152$$

$$\delta_3 = Ç_3 . (1 - Ç_3) . \delta_Ç . W_{35}(t-1)$$

$$\delta_3 = 0{,}584490 . (1 - 0{,}584490) . (-0{,}085459) . 0{,}164732 = -0{,}003418$$

$$\delta_4 = Ç_4 . (1 - Ç_4) . \delta_Ç . W_{45}(t-1)$$

$$\delta_4 = 0{,}471226 . (1 - 0{,}471226) . (-0{,}085459) . 0{,}752621 = -0{,}016026$$

$$\Delta W_{13} = 0{,}5 . \delta_3 . Ç_1 + 0{,}8 . \Delta W_{13}(t-1)$$

$$\Delta W_{13} = 0{,}5 . (-0{,}003418) . 0 + 0{,}8.0 = 0$$

$$\Delta W_{23} = 0{,}5 . \delta_3 . Ç_2 + 0{,}8 . \Delta W_{23}(t-1)$$

$$\Delta W_{23} = 0{,}5 . (-0{,}003418) . 0 + 0{,}8.0 = 0$$

$$\Delta W_{14} = 0{,}5 . \delta_4 . Ç_1 + 0{,}8 . \Delta W_{14}(t-1)$$

$$\Delta W_{14} = 0{,}5 . (-0{,}016026) . 0 + 0{,}8.0 = 0$$

$$\Delta W_{24} = 0{,}5 . \delta_4 . Ç_2 + 0{,}8 . \Delta W_{24}(t-1)$$

$$\Delta W_{14} = 0{,}5 . (-0{,}016026) . 0 + 0{,}8.0 = 0$$

$$\Delta \beta_j{}^{a} = \lambda . \delta_j . + \alpha . \Delta \beta(t-1)$$

$$\Delta \beta_3{}^{a} = 0{,}5 . \delta_3 . + 0{,}8 . \Delta \beta(t-1)$$

$$\Delta \beta_3{}^{a} = 0{,}5 . (-0{,}003418) . + 0{,}8.0 = -0{,}001709$$

$$\Delta \beta_4{}^{a} = 0{,}5 . \delta_4 . + 0{,}8 . \Delta \beta(t-1)$$

$$\Delta \beta_4{}^{a} = 0{,}5 . (-0{,}016026) . + 0{,}8.0 = -0{,}008013$$

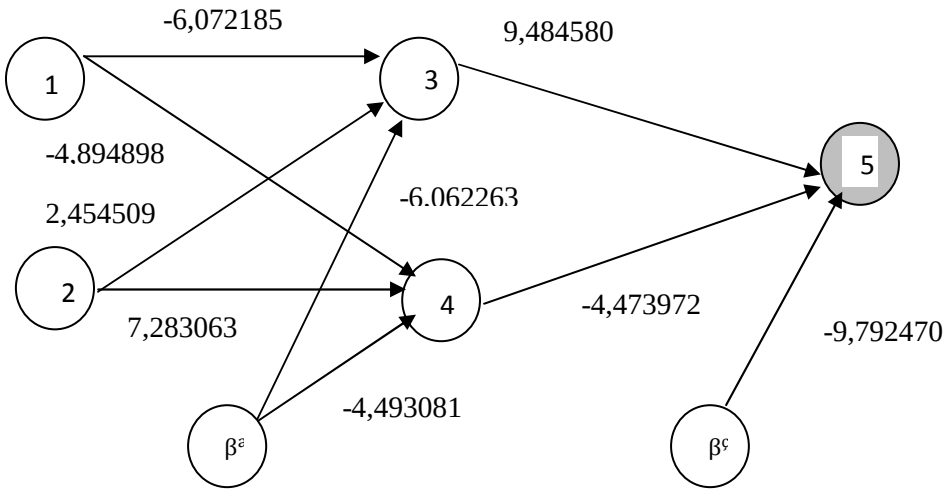
$N\beta_3{}^a = \beta_3{}^a + \Delta\beta_3{}^a$

$N\beta_3{}^a = 0{,}341332 - 0{,}001709 = 0{,}339623$

$N\beta_4{}^a = \beta_4{}^a + \Delta\beta_4{}^a$

$N\beta_4{}^a = -0{,}115223 - 0{,}008013 = 0{,}123236$

Birinci iterasyon bittikten sonra ikinci iterasyon başlar. Bu kez örnek2 ağa gösterilir. Yukarıdaki işlemler tekrar edilir, bütün çıktılar doğru cevap verinceye kadar iterasyonlar devam ettirilir. Bu problem için ağın öğrendikten sonraki değerleri aşağıdaki şekil 3.6. daki gibidir. Ayrıca tablo 3.2. de ağın XOR için sonuçları da gösterilmiştir.



Şekil 3.6. Ağın sonuç topolojisi

Tablo 3.2. Sonuçlar

	Girdi 1	Girdi 2	Beklenen Çıktı	Ağın çıktısı	Hata
Örnek 1	0	0	0	0,017620	~ 0,017
Örnek 2	0	1	1	0,981524	~ 0,018
Örnek 3	1	0	1	0,982492	~ 0,018
Örnek 4	1	1	0	0,022784	~ 0,020

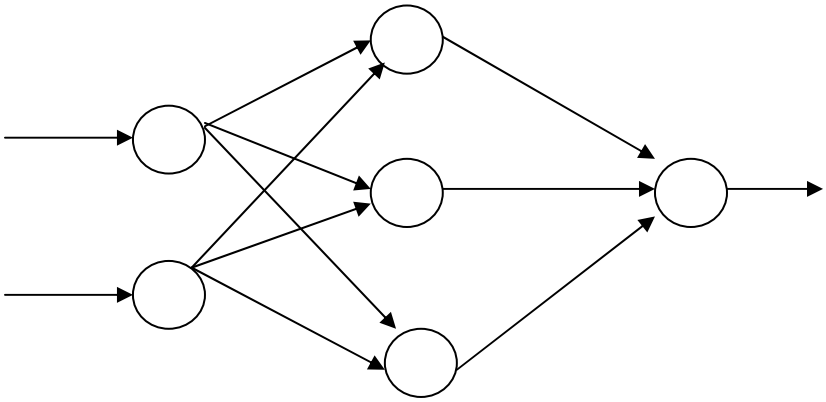
3.2.7. MATLAB ortamında yapay sinir ağının tasarımı

Yukarıda gösterilen örnekte görüldüğü üzere topolojik yapısı karmaşık olmayan bir problemde dahi hesaplama vakit alıcıdır. Bu nedenle zamanı etkin kullanma ve hesaplama açısından, yapay sinir ağ tasarım ve uygulamasını MATLAB ortamına geçirmek verimliliği arttıracaktır. Burada bir şeye dikkat çekilmesi gerekmektedir, o da bu uygulamayı yapmanın; YSA' nın temel mantığını, özelliklerini, çalışma prosedürünü bilmenin gereksiz olduğu anlamına kesinlikle gelmemesi olduğudur. Bunları bilerek bu uygulama etkin bir biçimde kullanılabilir.

Ağı oluşturma kodu
net = newff(PR,[S1 S2…SNl],{TF1 TF2…TFNl},BTF,BLF,PF)
PR: R elemanlı giriş vektörünün minimum ve maksimum değerlerini içeren Rx2 'lik matris olmak üzere

Si : i'nci katmanda bulunan nöron sayısı.
TFi : i'nci katmanın varsayılan transfer fonksiyonu,
BTF : Varsayılan geriye yayılım ağ eğitim fonksiyonu
BLF : Geriye yayılım ağırlık/bias öğrenme fonksiyonu
PF : Varsayılan performans fonksiyonu

Aşağıdaki kod ile 2 katmanlı bir ağ oluşturulsun. Bu problemde iki elamanlı bir giriş vektörü vardır. Giriş vektörünün ilk parametresinin değer aralığı -1 ve 2 olarak belirtilmiştir. İkinci parametrenin değer aralığı ise 0-5 olarak verilmiştir. Ara katmanda 3 nöron, çıkış katmanında ise 1 nöron bulunmaktadır. Ara katmanın transfer fonksiyonu sigmoid, ikinci katmanınki ise lineer'dir. Öğrenme algoritması, gradyent azaltım algoritmasıdır. Problem iki girişli bir çıkışlı bilinmeyen bir sistem olarak düşünülebilir. Amaç örnek giriş ve çıkış verileri vererek sistemi karakterize etmektir.



Şekil 3.7. Örnek Bir Ağ

net=newff([-1 2; 0 5],[3,1],{'logsig','purelin'},'traingd');

Yukarıda yazılan komut ağ nesnesi oluşturur ve aynı zamanda ağırlık ve bias değerlerinin ilklendirme (initialization) işlemini rastgele değerler atamak suretiyle gerçekleştirir. Haliyle ağ eğitime hazır olarak beklemektedir. Bazı durumlarda ağı tekrar ilklendirmek (resetlemek) gerekebilir. Bu durumda;
net = init(net); kodu kullanılır.

Tasarlanan ağın simülasyonu için sim komutu kullanılır. Simülasyon'dan kasıt, ağa bir giriş değeri verip ağın çıkışını hesaplatmaktır. Bu aşağıdaki kod ile gerçekleşir.

```
p = [1;2];
a = sim(net,p)
a =
  -0.1011
```

Burada p ağın giriş verisini temsil etmektedir. Ağın iki girişi olduğu için p matrisi iki satırdan meydana gelmektedir. Görüleceği üzere ağ bir çıkış üretmektedir. Ancak bu, ağ henüz eğitilmediği ve ağırlık değerleri rastgele atandığı için arzu edilen çıkış değildir. Bu komutlardan farklı bilgisayarlarda muhtemelen farklı bir çıkış elde edilecektir. Çünkü ağırlık değerleri ilklendirilirken rastgele olarak atanmaktadır. Diğer bir yandan ağ çoklu örneklerle de simüle edilebilir.

```
p = [1 3 2;2 4 1];
a=sim(net,p)
a =
  -0.1011   -0.2308    0.4955
```

Burada iki girişli ağ için 3'er adet değer verilmiştir. Haliyle ağ her bir örnek için bir çıkış üretmiştir. Buradan açıkça görülüyor ki ağın girişi için (simülasyon veya eğitim aşamasında) organize edilecek matrisin satırları parametrelere, sütunları ise örneklere karşılık gelmektedir.

Ağırlıklar ve bias değerleri ilklendirildikten sonra ağ eğitime hazırdır. Eğitim işleminde ağın davranışını ortaya koyacak giriş (p) ve çıkış (t) verileri gereklidir. Bunlar eğitim verisi olarak adlandırılır. Eğitim verileri çözüm uzayının tamamını veya en azından büyük bölümünü temsil edecek şekilde seçilmelidir. Eğitim esnasında ağırlık ve bias değerleri iteratif olarak hata fonksiyonunu minimize edecek şekilde güncellenir. YSA' da kullanılan en temel öğrenme algoritması geriye yayılım (back propagation) algoritmasıdır. Basit anlamda ağırlıklar negatif gradyent yönünde güncellenir. Toplu gradyent azaltım (batch gradient descent) eğitme algoritması MATLAB'de traingd ile temsil edilir [26].

Verdiğimiz problemin eğitim verilerini oluşturup ağın eğitimini şu şekilde gerçekleştirebiliriz:

```
p = [-1 -1 2 2;0 5 0 5];
t = [-1 -1 1 1];
```

Burada, p matrisi ağın eğitimi için kullanılacak giriş verisini temsil etmektedir. Görüldüğü üzere 2 satırdan ve 4 sütundan oluşmaktadır. Problem iki parametreli olduğu için satır sayısı 2'dir. Ağın eğitimi için ise 4 örnek verilmektedir. Ağın çıkışını t matrisi temsil etmektedir. Tek satır olduğuna göre tek çıkış parametresi vardır (hatırlanırsa sistemimiz 2 girişli 1 çıkışlıdır). Verilen sistemin doğruluğu tablo 3.3. teki gibidir.

Tablo 3.3. Doğruluk Tablosu

p_1	p_2	t
-1	0	-1
-1	5	-1
2	0	1
2	5	1

Ağın eğitimi ile ilgili temel parametreleri şu şekilde düzenlenebilir.

net.trainParam.show = 50;
net.trainParam.lr = 0.05;
net.trainParam.epochs = 300;
net.trainParam.goal = 1e-5;

Burada, show parametresi kaç iterasyonda bir eğitim durumunun MATLAB ekranına aktarılacağını belirmekte, epoch parametresi iterasyon sayısını, goal parametresi hedeflenen hata değerini (yukarıdaki örnek için 1e-5 = 10^{-5}) , lr ise geriye yayılım algoritması için öğrenme oranını belirtmektedir. epoch veya goal parametrelerinden birisi sağlandığında eğitim işlemi durdurulacaktır. Hata fonksiyonu genelde MSE olarak belirlenir, bu durumunda hata, arzu edilen çıkış (t) ile ağın o anki çıkışı arasındaki farkın karesinin ortalamasıdır.

Ağın eğitimine başlamak için train komutu kullanılır.

[net,tr]=train(net,p,t);
TRAINGD, Epoch 0/300, MSE 1.59423/1e-05, Gradient 2.76799/1e-10
TRAINGD, Epoch 50/300, MSE 0.00236382/1e-05, Gradient 0.0495292/1e-10
TRAINGD, Epoch 100/300, MSE 0.000435947/1e-05, Gradient 0.0161202/1e-10
TRAINGD, Epoch 150/300, MSE 8.68462e-05/1e-05, Gradient 0.00769588/1e-10
TRAINGD, Epoch 200/300, MSE 1.45042e-05/1e-05, Gradient 0.00325667/1e-10
TRAINGD, Epoch 208/300, MSE 9.64816e-06/1e-05, Gradient 0.00266775/1e-10
TRAINGD, Performance goal met.

Görüleceği üzere 208. iterasyonda hedeflenen hata oranına erişilmiş ve ağın eğitimi durdurulmuştur. Ağın eğitilip eğitilmediğini test etmek için giriş verisi (p) ile ağı tekrar simüle edilmeli:

```
a = sim(net,p)
a =
  -1.0010   -0.9989    1.0018    0.9985
```

Görüleceği üzere ağ arzu edilen çıkışları sağlamaktadır. Yani YSA verilen sistemi modellemiştir. Sistemin girişi ve çıkışı arasındaki bağıntıyı bulmuştur. Bunu da uygun ağırlık değerlerini seçerek yapmıştır. Bu aşamadan sonra eğitim verisinde kullanılmayan yeni veriler verilerek ağ test edilebilir.

MATLAB' da kullanabileceğimiz pek çok eğitim algoritması bulunmaktadır. En temel algoritma yukarda bahsedilen geriye yayılım tabanlı toplu gradyent azaltım algoritmasıdır. Fakat MATLAB'de varsayılan eğitim algoritması olarak Levenberg-Marquardt (trainlm) seçilmiştir. Bu algoritma çok hızlıdır ancak çalışabilmesi için fazla hafızaya ihtiyaç duymaktadır. Bunların yanı sıra trainbfg ve trainrp algoritmaları da tercih edilebilir.

Ön-işleme seçenekleri (Preprocessing)

Bazı temel ön işlemleri gerçekleştirerek YSA' nın eğitim ve öğrenme performansı artırılabilir. En temel ön-işleme yöntemlerden birisi giriş verilerinin normalize edilmesidir. premnmx komutu ile giriş ve çıkış verilerini -1 ile 1 arasında ölçeklemek mümkündür.

```
[pn,minp,maxp,tn,mint,maxt] = premnmx(p,t);
net = train(net,pn,tn);
```

Eğitimden sonra simulasyon aşamasında ise normalize edilen verilerin tekrar eski haline çevrilmesi gerekmektedir. Bu işlem için postmnmx komutu kullanılır.

```
an = sim(net,pn);
[a] = postmnmx(an,mint,maxt);
```

Bir diğer normalizasyon yöntemi de verilerin ortalamasını 0 ve standart sapmasını -1 olacak şekilde yeniden düzenlemektir. Bu amaç için prestd ve poststd komutları kullanılmaktadır.

```
p = [-0.92 0.73 -0.47 0.74 0.29; -0.08 0.86 -0.67 -0.52 0.93];
t = [-0.08 3.4 -0.82 0.69 3.1];

[pn,meanp,stdp,tn,meant,stdt] = prestd(p,t);
net = newff (minmax(pn), [5 1],{'tansig' 'purelin'},'trainlm');
net = train (net,pn,tn);
an = sim (net,pn);
a = poststd (an,meant,stdt);
```

Bazı durumlarda ağın girişi çok fazladır. Bu durumlarda eğitim işlemi çok uzun sürmektedir. Eğer giriş verileri arasında birbirleri ile ilintili veriler varsa bunlar ağın eğitimi için bir katkıda bulunmayacaktır. Temel bileşen analizi (principle component analysis, PCA) adı verilen teknikle ağın eğitimine katkıda bulunmayacak veriler indirgenerek giriş verisi azaltılabilir. Böylece giriş verilerinin kendi aralarında korelasyonlu olması engellenmiş olur.

```
p=[-1.5 -0.58 0.21 -0.96 -0.79; -2.2 -0.87 0.31 -1.4  -1.2];

[pn,meanp,stdp] = prestd(p);

[ptrans,transMat] = prepca(pn,0.02);
```

Buradaki örnekte giriş verisinin ikinci satırı birinci satırın yaklaşık olarak belli bir katsayı ile çarpılmış halidir. Temel bileşen analizi sonucunda iki parametreli giriş vektörü tek parametreli hale getirilmiştir (ptrans).

Örnek: EXOR çözmek için MLP egitimi

```
%% EXOR çözmek için MLP egitimi
clear all; close all; clc;
x=[0 0 1 1;0 1 0 1];
yd=[0 1 1 0];
net=newff(minmax(x),[2 1],{'logsig','logsig'},'traingd');
% net=newff(minmax(x),[2 1],{'logsig','logsig'},'traingdm');
% net=newff(minmax(x),[2 1],{'logsig','logsig'},'traingda');
% net=newff(minmax(x),[2 1],{'logsig','logsig'},'traingdx');
% net=newff(minmax(x),[2 1],{'logsig','logsig'},'trainrp');
% net=newff(minmax(x),[2 1],{'logsig','logsig'},'traincgf');
% net=newff(minmax(x),[2 1],{'logsig','logsig'},'traincgp');
% net=newff(minmax(x),[2 1],{'logsig','logsig'},'traincgb');
% net=newff(minmax(x),[2 1],{'logsig','logsig'},'trainscg');
% net=newff(minmax(x),[2 1],{'logsig','logsig'},'trainbfg');
% net=newff(minmax(x),[2 1],{'logsig','logsig'},'trainoss');
% net=newff(minmax(x),[2 1],{'logsig','logsig'},'trainlm');
W1=[-1.5804 -0.6817; % baslangıç parametreleri
-0.0787 -1.0246];
W2=[-1.2344 0.2888];
b1=[-0.4293;0.0558];
b2=-0.3679;
net.iw{1}=W1;
net.lw{2}=W2;
net.b{1}=b1;
net.b{2}=b2;
net.trainParam.epochs=3000;
net.trainParam.show = 100;
net.trainParam.lr=0.5;
net.trainParam.mc = 0.5;
net.trainParam.goal=0.001;
[net,tr]=train(net,x,yd);
y = sim(net,x)
```

Çarpma işleminin ysa' ya öğretilmesi

Çarpma işlemi doğrusal olmayan bir işlemdir ve yapay zekâ için zor bir problemdir. Bu örnekte 1 ile 10 arasındaki sayıların çarpım işlemi YSA'ya öğretilmiştir. Çarpma işleminde iki giriş ve bir çıkış vardır. Eğitimde 20 adet rastgele sayı seçilmiş ve test aşamasında yine 20 adet rastgele sayı seçilerek sistemin performansı oraya konmuştur.

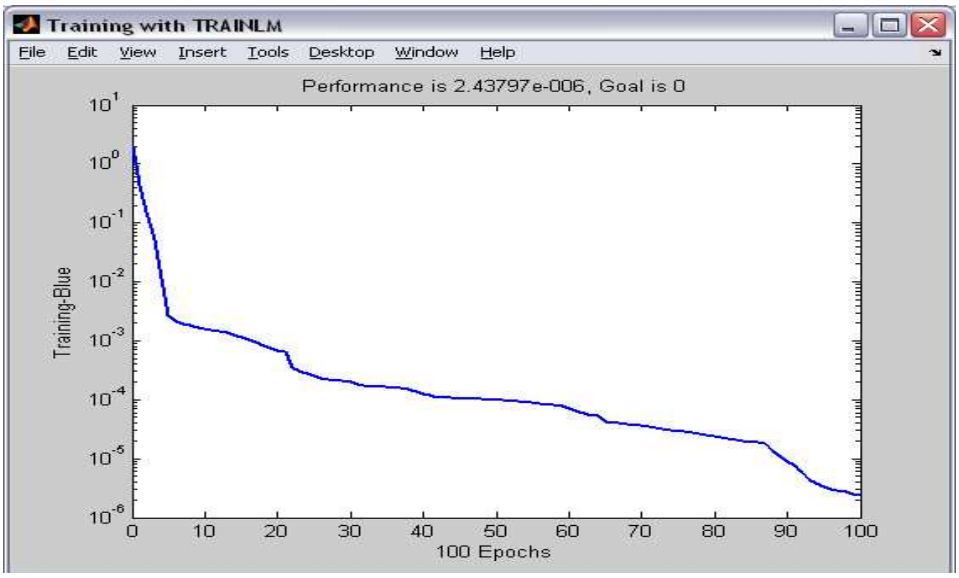
Örnek:

```
clear all, close all,clc;
p=round(rand(2,20)*10) % Giris verisi 2x20'lik matris, rastgele 0-10
t = p(1,:).*p(2,:)     % Cikis verisi p1*p2
%Normalizasyon islemi
[pn,minp,maxp,tn,mint,maxt] = premnmx(p,t);
%YSA'nin tasarimi, egitimi ve simulasyonu
net = newff(minmax(pn),[5 1],{'tansig' 'purelin'},'trainlm');
net = train(net,pn,tn);
an = sim(net,pn);
[a] = postmnmx(an,mint,maxt); %Normalizasyonun tersi
%Egitim verilerinin gercek ve YSA cikisinin gosterimi
figure(1),plot3(p(1,:),p(2,:),t,'o');
hold on,plot3(p(1,:),p(2,:),a,'r*'),grid on;
legend('Gerçek deger','YSA cikisi'),xlabel('p1'),ylabel('p2'),zlabel('t'),title('Egitim
verisi')
%Test verilerinin hazirlanmasi, farkli 20 adet ornek
ptest=round(rand(2,20)*10)
ttest = ptest(1,:).*ptest(2,:)
[ptn,minpt,maxpt,ttn,mintt,maxtt] = premnmx(ptest,ttest);
atn = sim(net,ptn); %Simulasyon
[at] = postmnmx(atn,mintt,maxtt);
%Test verilerinin gercek ve YSA cikisinin gosterimi
figure(2),plot3(ptest(1,:),ptest(2,:),ttest,'o');
```

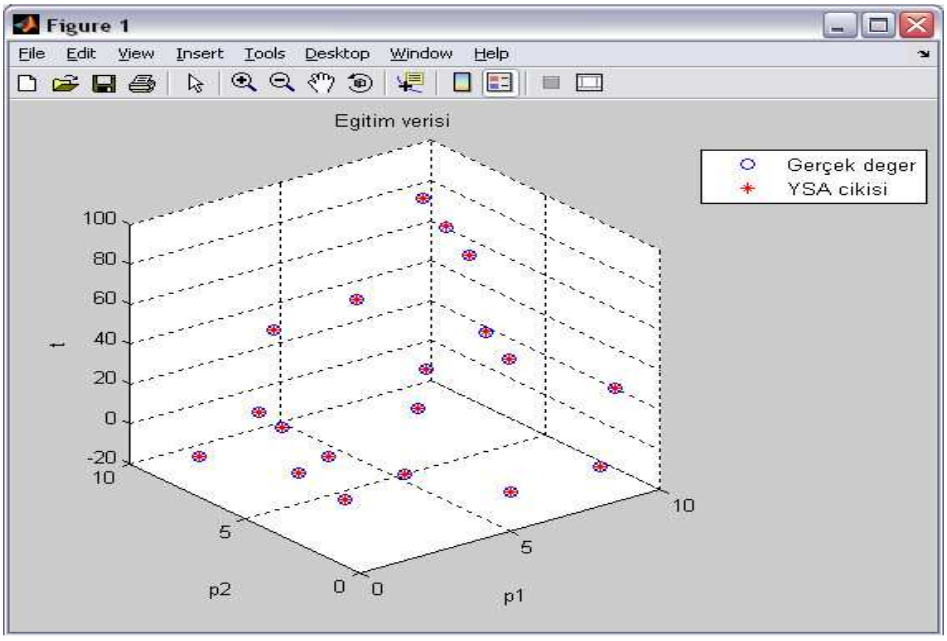
hold on,plot3(ptest(1,:),ptest(2,:),at,'r*'),grid on;

legend('Gerçek deger','YSA cikisi'),xlabel('p1'),ylabel('p2'),zlabel('t'),title('Test verisi')

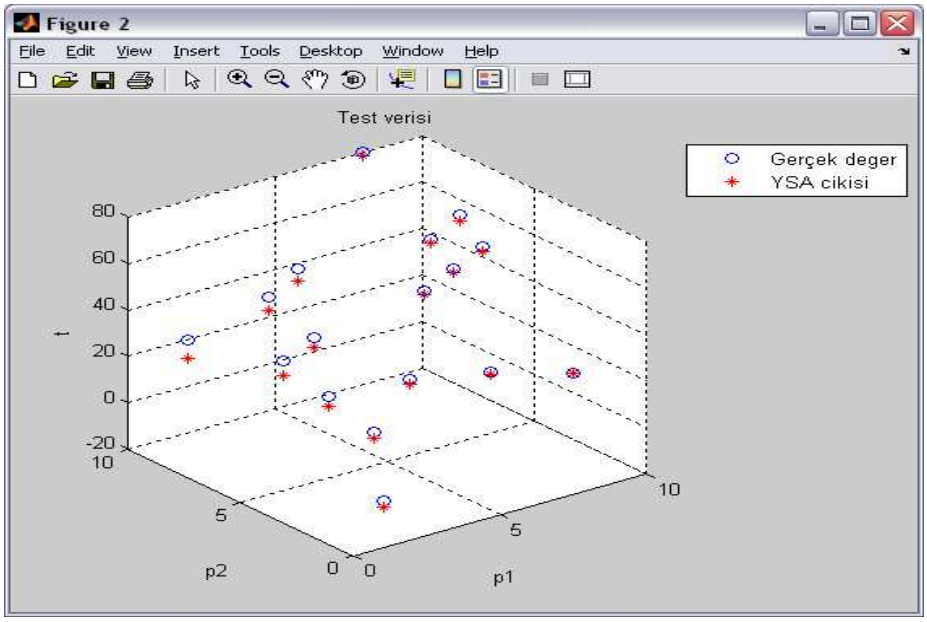
Programın ekran çıktıları aşağıdaki şekillerde gösterilmiştir şekil 3.8. şekil 3.9. ve şekil 3.10. da gösterilmiştir.



Şekil 3.8. Eğitim Ekran Çıktısı



Şekil 3.9. Eğitim Gerçek Değer ve YSA Çıktısı



Şekil 3.10. Gerçek Değer ve YSA Çıktısı

Geri yayılım öğrenme ağı örneklendirilmesi

Levenberg-Marquardt hızlı yakınsamasıyla bilenen bir yapay sinir ağı eğitim yöntemidir. Levenberg-Marquardt öğrenim algoritması hızlı öğrenme amaçlı kullanılmakta olup Hessian matrisini hesaplamadan sonuca varmaya çalışmaktadır. Performans fonksiyonu karelerin toplamı (ağın ileri destekli (feedforward) öğrenimi) formunda olursa Hessian matrisi aşağıdaki gibi kestirilebilir.

$$H = J^T.J$$

ve yakınsama eğilimi (gradyan)

$$g = J^T.e$$

şeklinde hesaplanır.

Denklemlerdeki " J " nonlineer optimizasyonda da bahsedildiği üzere jacobian matrisi; öğrenim sürecindeki hataların, ağdaki ağırlıklara ve eğime göre türevini ifade eder. " e " ise ağdaki hatalar vektörüdür.

Bunun için Levenberg-Marquardt geri yayılım öğrenme ağlarında (Levenberg-Marquardt backpropagation), öğrenme performansının Jacobian jx'i hesaplanmaktadır. Ağdaki her değişkenin değeri Levenberg-Marquardt kurallarına göre değiştirilir. Matlab'daki " trainlm " fonksiyonu ağın öğrenimi için kullanılmaktadır, " trainlm " fonksiyonu ağ ara katmanındaki ağırlık değerlerinin, ağ girdilerinin ve transfer fonksiyonlarının türevleri olduğu sürece yakınsama işlemini sürdürmektedir [38].

Bu yöntemde ağ parametrelerinin deneme yanılma yöntemi ile en makul parametreleri belirlenmesi gerekmektedir. Ağın eğitimi aşağıdaki koşullardan herhangi birisi oluştuğunda durdurulmaktadır:

Tekrar sayısının üst sınırına ulaşıldığı zaman (epochs).
Ağ koşturma zamanının üst sınırına ulaşıldığı zaman.
Performans hedeflenen değere ulaştığı zaman
Performans gradyanı (yaklasımı, yakınsaması) belirlenen alt değerin (mingrad) altına düştüğü zaman eğitim durdurulur.

Yapay sinir ağını oluşturmak, eğitmek ve test ederek başarı oranlarını bulmak için yazılan MATLAB kodu ve kod açıklaması aşağıda verilmektedir:

```
1.  pred = 0; MMRE = 0; mse = 0; gtmp = 0; y=0; yp=0;
2.  input=[data(:,2)'; data(:,3)';data(:,4)'; ];
3.  target=[data(:,1)'];
4.  [input,minp,maxp,target,mint,maxt] = premnmx(input,target);
5.  net=newff(minmax(input),[10,1],{'logsig','purelin'},'trainlm');
6.  net.trainParam.goal=1e-7;
7.  net.trainParam.epochs=10000;
8.  [net,tr]=train(net,input,target);
9.  for i = 1 : 15
10. y = testdata(i,2:4);
11. y=tramnmx(y',minp,maxp);
12. output = sim(net,y);
13. yp = postmnmx(output,mint,maxt);
14. mse = abs((testdata(i,1)-yp)./testdata(i,1));
15. if mse <= 0.25
16. pred = pred + 1;
17. end;
18. gtmp = gtmp + mse;
19. end;
20. MMRE = gtmp / 15;
21. disp('pred='); disp(pred/15);
22. disp('MMRE='); disp(MMRE);
23. end;
```

1. pred = 0; MMRE = 0; mse = 0; gtmp = 0; y=0; yp=0;

Bu komutlarla, kullanılan değişkenlerin ilk değerleri verilmektedir.

2. input=[data(:,2)'; data(:,3)';data(:,4)';];

Bu komut sonunda input matrisi, 100 tane eğitim verisini içermektedir. Verilerin (data) ilk sütunu maliyet cinsinden efordur yani yapay sinir ağının çıktısını oluşturacağından kullanılmayacaktır. Matlab ortamında Yapay Sinir Ağları'ndaki bir girdiye ilişkin değerlerin satırlara yerleştirilmesi gerektiğinden, 2, 3, ve 4. sütunların transpozları alınmıştır. Bu durumda girdi matrisimiz 3 satır (ağın girdisi) ve 100 sütundan oluşmaktadır

3. target=[data(:,1)'];

Yapay sinir ağlarının eğitiminde kullanılan maliyet verilerimiz, data matrisinin 1. sütununda bulunduğundan transpozu alınarak target dizisine yerleştirilmiştir. Bu dizi maliyet cinsinden gerçek eforları tutmaktadır.

4. [input,minp,maxp,target,mint,maxt] = premnmx(input,target);

Bu komut, veriler üzerinde ön işlem uygulamaktadır. Yapay Sinir ağlarına girdi ve çıktı olarak verilecek değerler, farklı tiplerde olduklarından, çok değişik ölçeklerde olabilmektedirler. Hepsinin -1 ile +1 aralığına getirilmesi, ağın etkin bir şekilde öğrenebilmesi için gerekmektedir.

5. net=newff(minmax(input),[10,1],{'logsig','purelin'},'trainlm');

Bu komut ile, ileri sürümlü çok katmanlı algılayıcı yapay sinir ağı oluşturulmaktadır. Ara katmanda bu örnek için 10 adet nöron bulunmaktadır. Ara katman aktarım fonksiyonu, genelde önerildiği gibi logsig olarak belirlenmiştir. Çıktı katmanındaki nöron sayısı da, tek sonuç (maliyet) tahminlendiğinden, 1 olarak verilmiştir. Levenberg-Marquardt geri yayılım öğrenme algoritması kullanılmıştır.

6. net.trainParam.goal=1e-7;

Hata hedefimiz,10^{-7} olarak belirtilmiştir. Bu hata değerine ulaşılıncaya kadar, tüm eğitim verileri yapay sinir ağına tekrar tekrar verilmektedir. Bu değere ulaşıldığında, eğitim tamamlanmıştır ve sonlandırılmaktadır.

7. net.trainParam.epochs=10000;

Eğitimin ne kadar süreceği belirtilmektedir. Hata hedefine daha önce ulaşamazsa, eğitim verisi ağa 10000 kere verilecektir. Daha büyük değerler verildiğinde, eğitim süresi uzar ancak ağ daha iyi öğrenir.

8. [net,tr]=train(net,input,target);

Önceki satırlarda atamalarını yaptığımız girdi ve gerçek çıktı değerlerine göre, oluşturduğumuz ağ üzerinde eğitim işlemini gerçekleştirir. Ağ üzerinde, katmanlar arasındaki tüm ağırlıklar, öğrenme kalitesine göre ayarlanmaktadır.

```
 9. for i = 1 : 15
10. y = testdata(i,2:4);
11. y = tramnmx(y',minp,maxp);
12. output = sim(net,y);
13. yp = postmnmx(output,mint,maxt);
14. mse = abs((testdata(i,1)-yp)./testdata(i,1));
15. if mse <= 0.25
16. pred = pred + 1;
17. end;
18. gtmp = gtmp + mse;
19. end;
```

Bu satırlarda, eğitim verisinden ayrı olarak belirlediğimiz 15 adet test verisi, tramnmx komutu ile önişlemden geçirilmektedir. Döngü her bir test verisi için tekrarlanmaktadır. sim komutu ile, önceden eğitim verisi ile eğitmiş olduğumuz ağın ağırlıklarına göre output değişkenine tahminlenmiş maliyet değeri atanmaktadır. Ardından, elde ettiğimiz sonuç (maliyet degeri), [-1,1] aralığından, orijinal maliyet değeri ölçeğine ayarlanacak şekilde son işlemden geçirilmektedir. Bu değer ile gerçek hayattaki maliyet arasındaki fark, yine gerçek değere bölünerek mse değeri elde edilmektedir. Bu değer 0.25'ten küçükse sayacımızı bir artırıyoruz. İç döngünün bitiminden sonra, gtmp değerine, ilgili test verisinin hesapladığımız mse değerini ekliyoruz.

```
20. MMRE = gtmp / 15;
21. disp('pred='); disp(pred/15);
22. disp('MMRE='); disp(MMRE);
23. end;
```

Bu satırlarda MMRE değerini, hesapladığımız MSE toplamından yararlanarak hesaplıyoruz. Başarı oranını (pred) ve MMRE (ağın göreceli hata deri) değerini ekrana yazdırılır.

3.3. Genetik algoritmalar

Genetik algoritma, büyük ve lineer olmayan arama uzaylarında geleneksel hesaplama yöntemlerinin işe yaramadığı durumlarda kullanılabilen doğal seleksiyon ve doğal genetiğin yöntemlerini kullanan bir arama algoritmasıdır. Genetik algoritma, bir veri grubundan özel bir veriyi bulmak için kullanılır. Genetik algoritmalar 1970'lerin basında John Holland tarafından ortaya atılmıştır. Genetik Algoritmalar, Evrimsel Genetik ve Darwin'in Doğal seleksiyonuna benzerlik kurularak geliştirilmiş " iteratif (ardışık) ", ihtimali bir arama metodudur [30].

" GA Darvin'in evrim teorisinin doğal seçilim ilkesinden esinlenerek oluşturulmuştur. Herhangi bir problemin genetik algoritma ile çözümü, problemin sanal olarak evrimden geçirilmesi ile gerçekleştirilmektedir. " [34]. Genetik algoritmalar bir çözüm uzayındaki her noktayı, kromozom adı verilen ikili bit dizisi ile kodlar. Her noktanın bir uygunluk değeri vardır. Tek bir nokta yerine, genetik algoritmalar bir populasyon olarak noktalar kümesini muhafaza eder. Her kuşakta, genetik algoritma, çaprazlama ve mutasyon gibi genetik operatörleri kullanarak yeni bir populasyon oluşturur. Birkaç kuşak sonunda, populasyon daha iyi uygunluk değerine sahip üyeleri içerir. Bu, Darwin'in rastsal mutasyona ve doğal seçime dayanan evrim modellerine benzemektedir. Genetik algoritmalar, çözümlerin kodlanmasını, uygunlukların hesaplanmasını, çoğalma, çaprazlama ve mutasyon operatörlerinin uygulanmasını içerir [31].

GA doğada geçerli olan en iyinin yaşaması (Survival of the Fittest) kuralına dayanarak sürekli iyileşen çözümler üretir. Bunun için " iyi "nin ne olduğunu belirleyen bir uygunluk (fitness) fonksiyonu ve yeni çözümler üretmek için yeniden kopyalama (recombination), mutasyon (mutation) gibi operatörleri kullanır. Genetik algoritmaların bir diger önemli özelliği de bir grup çözümle uğraşmasıdır. Bu sayede çok sayıda çözümün içinden iyiler seçilip kötüler ise elenebilir.

Problemin bireyler içindeki gösterimi problemden probleme değişiklik gösterir. Genetik algoritmaların problemin çözümündeki başarısına karar vermedeki en önemli faktör, problemin çözümünü temsil eden bireylerin gösterimidir.

Popülasyon içindeki her bireyin problem için çözüm olup olmayacağına karar veren bir uygunluk fonksiyonu vardır. Uygunluk fonksiyonundan dönen değere göre yüksek değere sahip olan bireylere, popülasyondaki diğer bireyler ile çoğalmaları için fırsat verilir. Bu bireyler çaprazlama işlemi sonunda çocuk adı verilen yeni bireyler üretirler. Çocuk kendisini meydana getiren ebeveynlerin (anne, baba) özelliklerini taşır. Yeni bireyler üretilirken düşük uygunluk değerine sahip bireyler daha az seçileceğinden bu bireyler bir süre sonra popülasyon dışında bırakılırlar. [31-33].

Yeni popülasyon, bir önceki popülasyonda yer alan uygunluğu yüksek bireylerin bir araya gelip çoğalmalarıyla oluşur. Aynı zamanda bu popülasyon önceki popülasyonun uygunluğu yüksek bireylerinin sahip olduğu özelliklerin büyük bir kısmını içerir. Böylelikle pek çok nesil aracılığıyla iyi özellikler popülasyon içersinde yayılırlar ve genetik işlemler aracılığıyla da diğer iyi özelliklerle birleşirler. Uygunluk değeri yüksek olan ne kadar çok birey bir araya gelip, yeni bireyler oluştururursa arama uzayı içerisinde o kadar iyi bir çalışma alanı elde edilir.

Genetik algoritmalarda, probleme ait en iyi çözümün bulunabilmesi için;

Bireylerin gösterimi doğru bir şekilde yapılmalı,
Uygunluk fonksiyonu etkin bir şekilde oluşturulmalı,
Doğru genetik işlemciler seçilmelidir.

Bu durumda çözüm kümesi problem için bir noktada birleşecektir. GA diğer optimizasyon yöntemleri kullanılırken büyük zorluklarla karşılaşılan, oldukça büyük arama uzayına sahip problemlerin çözümünde basarı göstermektedir. Bir problemin bütünsel en iyi çözümünü bulmak için garanti vermezler. Ancak problemlere makul bir süre içinde, kabul edilebilir, iyi çözümler bulurlar. GA nın asıl amacı, hiçbir çözüm tekniği bulunmayan problemlere çözüm aramaktır [34].

GA ancak;

Arama uzayının büyük ve karmaşık olduğu,
Mevcut bilgiyle sınırlı arama uzayında çözümün zor olduğu,
Problemin belirli bir matematiksel modelle ifade edilemediği,
Geleneksel optimizasyon yöntemlerinden istenen sonucun alınmadığı alanlarda etkili ve kullanışlıdır [34].

Genetik algoritmaları diğer algoritmalardan ayıran en önemli özelliklerden biri de seçilimdir. Genetik algoritmalarda çözümün uygunluğu onun seçilme sansını arttırır ancak bunu garanti etmez. Seçim de ilk grubun oluşturulması gibi rastgeledir ancak bu rastgele seçimde seçilme olasılıklarını çözümlerin uygunluğu belirler.

3.3.1. Tarihçe ve uygulama alanları

Evrimsel hesaplamanın (Evolutional Computing) en çok kullanılan yaklaşımı olan genetik algoritmalar, 1960'larda John Holland tarafından bulundu. Adaptation in Natural and Artificial Systems (Doğal ve Yapay Sistemlerde Adaptasyon) adlı kitabında Holland GA'yı doğadaki adaptasyondaki teorik çerçevede açıkladı. Holland'ın çıkış noktası oldukça bilimseldi. Doğadaki değişik fenomenleri anlamaya ve onlarla bağlantılar kurmaya çalışırken aynı zamanda potansiyel mühendislik uygulamaları önermekteydi. Holland'ın kitabının yayınlanmasından sonra GA yapay zeka ve makine öğrenmesi konularında büyük bir alt alan oldu.

1985 yılında Holland'ın öğrencisi olarak doktora yapan David E. Goldberg adlı inşaat mühendisi 1989'da konusunda bir klasik sayılan kitabını yayınlayana kadar genetik algoritmaların pek pratik yararı olmayan bir araştırma konusu olduğu düşünülüyordu. Goldberg, GA'nın çok sayıda kollara ayrılmış gaz borularında, gaz akışını düzenlemek ve kontrol etmek için bir uygulamasını yapmıştır. Goldberg'in gaz boru hatlarının denetimi üzerine yaptığı doktora tezi ona sadece 1985 National Science Foundation Genç Araştırmacı ödülünü kazandırmakla kalmadı, genetik algoritmaların pratik kullanımının da olabilirliğini kanıtladı. Ayrıca kitabında genetik algoritmalara dayalı tam 83 uygulamaya yer vererek GA'nın dünyanın her yerinde

çeşitli konularda kullanılmakta olduğunu gösterdi. Günümüzde genetik algoritma, uzman sistemler ve yapay sinir ağları ile birlikte yapay zeka uygulamalarının ana araçlarından biri haline gelmiştir.

Genetik algoritmalar biyoloji, kimya, bilgisayar bilimleri ve sosyal bilimler gibi birçok alana uygulanmıştır. Genetik algoritmaların en uygun olduğu problemler geleneksel yöntemler ile çözümü mümkün olamayan ya da çözüm süresi problemin büyüklüğü ile üstel orantılı olarak artan (NP-hard) problemlerdir. Bir arama yöntemi olan genetik algoritmalar, farklı bilim dallarındaki optimizasyon problemlerini çözmede kullanılmaktadır.

Genetik algoritmaların uygulandığı optimizasyon problemleri, fonksiyon optimizasyonu ve birleşi (combinatorial) optimizasyonu altında toplanabilir. Genetik algoritma araştırmalarının önemli bir bölümü fonksiyon optimizasyonu ile ilgilidir. Genetik algoritmalar, geleneksel optimizasyon tekniklerine göre zor, süreksiz ve gürültü (noisy) içeren fonksiyonları çözmede daha etkindirler [33].

Optimize edilecek amaç fonksiyonunun süreksiz olması halinde, süreksizlik noktalarında fonksiyonun türevi alınamayacağından, türev almaya dayalı optimizasyon yöntemleri kullanılamamaktadır. Oysa, genetik algoritmalar, problemlerin çözümü için türev veya diğer yardımcı bilgilere gereksinim yöntemlere göre önemli bir üstünlük sağlamaktadır.

Genetik algoritmaların uygulandığı diğer bir optimizasyon problem sınıfı olan birleşi optimizasyon problemleri ise, istenen amaçlara ulaşmak üzere, sınırlı kaynakların etkin tahsis edilmesiyle ilgilidir. Bu sınırlar genel olarak, işgücü, tedarik veya bütçe ile ilgilidir. Sözü geçen “birleşi” kelimesi, yalnızca sonlu sayıda alternatif uygun çözümün mevcut olması ile ilgilidir. Birleşi optimizasyon, iyi tanımlanmış bir problem uzayında bir veya daha fazla optimal çözüm bulma sürecidir. Bu tip problemler yönetim biliminin tüm dallarında da (finans, pazarlama, üretim, stok kontrolü, veri-tabanı yönetimi vb.) ortaya çıkmaktadır. Gezgin satıcı problemi, araç rotalama problemi, Çinli postacı problemi, iş atölyesi çizelgeleme problemi, atama problemi, yerleşim tasarımı problemi ve sırt çantası problemi birleşi optimizasyon

problemlerine örnektir Birleşi optimizasyon problemlerinde, incelenen değişken sayısı arttıkça çözüme ulaşma zamanı üstsel olarak artmaktadır. Çözüm uzayının tamamının taranmasını gerektiren geleneksel çözüm yöntemlerinde problem çözümü değişken sayısının artmasıyla imkansız hale gelebilmektedir. Genetik algoritmalar ise çözüm uzayının yalnızca belirli bir kısmını taradığı ve eş zamanlı arama yaptığı için, bu tip problemlerde çözüme daha kısa sürede ulaşabilmektedir [31].

GA, yapay sinir ağları yaklaşımlarıyla birlikte kullanılmaktadır. Yumuşak hesaplama ve hibrid genetik algoritma yaklaşımı sık görülmektedir Ayrıca, çözüm performansı açısından finans problemlerindeki genetik algoritma çözümleri yasaklı arama, tavlama benzetimi arama metotları ile karşılaştırılmakta ve o probleme uygun çözüm yöntemi önerilmektedir. Genetik algoritmaların optimal kaynak tahsisi problemlerine uygulanması ile ortalama-varyans optimumundan farklı çözüm yöntemi geliştirilmiş ve kuadratik optimizasyona genetik algoritmalar uygulanmış olmaktadır.

Çeşitli avantajlarına rağmen GA uygulamalarında bir takım sorunlarla da karşılaşılmaktadır. Bu sorunları aşmak için çeşitli yöntemler geliştirilmiştir. Buna kısıtların ele alınmasındaki soruna karşı ceza fonksiyonu yönteminin kullanılması örnek verilebilir. Ancak, bulunan çeşitli yöntemlere rağmen bu konuda yeni yaklaşımlara gereksinim duyulmaktadır.

3.3.2. Temel kavramlar

Genetik algoritma tabanlı algoritmaların geliştirilmesindeki temel konular kromozom gösterimleri (kodlama), başlangıç popülasyonunun oluşturulması, evrim ölçüsü (uygunluk), çaprazlama, mutasyon ve seleksiyon (seçilim) stratejisidir. Bahsedilen çözümler kümesine popülasyon denir. Bu popülasyonun her bir elemanı bir birey ya da kromozom olarak tanımlanır ve bu birey çözümün kodlanmış bir biçimidir.

- **Kodlama (bireylerin gösterimi)**

Bir problemin çözümü için genetik algoritma geliştirmenin ilk adımı, tüm çözümlerin aynı boyutlara sahip bitler dizisi biçiminde gösterilmesidir. Dizilerden her biri, problemin olası çözümler uzayındaki rastsal bir noktayı simgeler. Parametrelerin kodlanması, probleme özgü bilgilerin genetik algoritmanın kullanacağı şekle çevrilmesine olanak tanır [31].

Herhangi bir evrimsel algoritmanın ilk adımı problemde aday sonucun nasıl gösterileceğine karar verilmesidir. Bu adım genotipin belirlenmesi ve genotipin fenotipe haritalanmasını içerir [34]. Kodlamanın amacı tıpkı biyolojideki genetikte olduğu gibi, bir çözümün kromozomu meydana getirecek genler şekline sokulmasıdır. Kodlama planı Genetik algoritmanın önemli bir kısmını teşkil eder. Çünkü bu plan bilginin çerçevesini şiddetle sınırlayabilir. Öyle ki probleme özgü bilginin bir kromozomsal gösterimiyle temsili sağlanır.

Kromozom genellikle, problemdeki değişkenlerin belli bir düzende sıralanmasıdır. Kromozomu oluşturmak için sıralanmış her bir değişkene “gen” adı verilir. Buna göre bir gen kendi başına anlamlı genetik bilgiyi taşıyan en küçük genetik yapıdır. Mesela; 101 bit dizisi bir noktanın x-koordinatının ikilik düzende kodlandığı gen olabilir. Aynı şekilde bir kromozom ise bir ya da daha fazla genin bir araya gelmesiyle oluşan ve problemin çözümü için gerekli tüm bilgiyi üzerinde taşıyan genetik yapı olarak tanımlanabilir. Örnek vermek gerekirse; 100011101111 x1, y1, x2, y2 koordinatlarından oluşan iki noktanın konumu hakkında bize bilgi verecektir. Bu parametreleri kodlarken dikkat edilmesi gereken en önemli noktalardan biri ise kodlamanın nasıl yapıldığıdır. Kodlama stratejisi, her optimizasyon problemi için farklıdır.

Örnek olarak kimi zaman bir parametrenin doğrusal ya da logaritmik kodlanması GA performansında önemli farka yol açar. Kodlamanın diğer önemli bir hususu ise kodlama gösteriminin nasıl yapıldığıdır. Bu da yeterince açık olmamakla birlikte GA performansını etkileyen bir noktadır. Problemin çözülebilmesi için doğru gösterimin seçilmesi önemlidir. İyi bir evrimsel algoritma tasarlamanın en zor kısmı gösterimi

doğru yapmaktır. Bu genelde pratikle ve uygulama alanını iyi tanımakla sağlanır [34].

Permütasyon kodlama

Gezgin satıcı problemi veya iş sıralama problemi gibi düzenleme problemlerinde kullanılır. Burada her kromozom sayıları bir sırada temsil eden bir sayı katarıdır.

Tablo 3.4. Permütasyon Kodlama

Kromozom A	3 5 7 4 2 6 8 9 1
Kromozom B	6 7 2 5 8 9 4 1 2

Değer kodlama

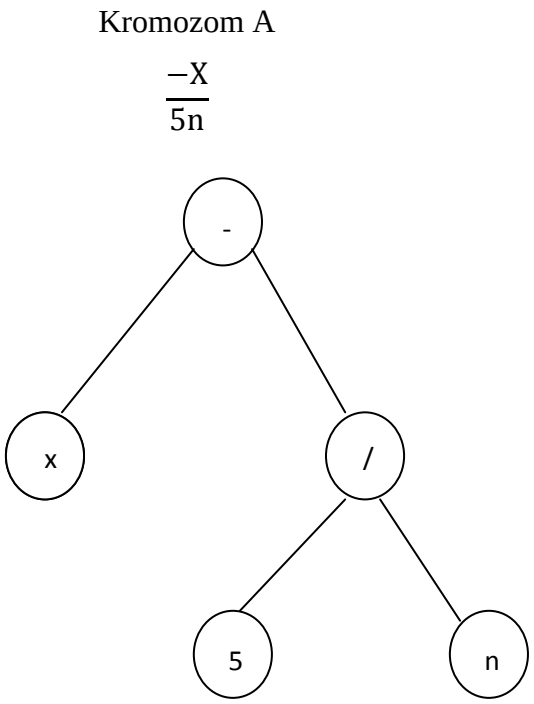
Reel sayılar gibi karmaşık değerlerin kullanıldığı problemlerde direk değer kodlanması kullanılabilir. Bu tip problemler için ikilik sistem (binary) kodlama çok zordur ve kullanışlı değildir.

Ağaç kodlama

Bu yöntem, genetik kodlama gibi gelişen ve değişen programlar veya ifadeler için kullanılır. Ağaç kodlamada her kromozom bazı nesnelerin, mesela fonksiyonlar ya da programlama dilindeki komutlar gibi, bir ağacıdır.

Tablo 3.5. Ağaç Kodlama

Kromozom X	1,3225 2,4673 3,1257 2,8976
Kromozom Y	ADEFCDABBCADFEGLT
Kromozom Z	right left back right left left



Şekil 3.11. Ağaç Kodlama Örneği

- **Uygunluk (Uyum)**

Başlangıç topluluğu bir kez oluşturulduktan sonra evrim başlar. GA bireylerin uygunluk ve iyiliklerine göre ayrılıp fark edilmesine gerek duyar. Uygunluk, topluluktaki bir kısım bireyin problemi nasıl çözeceği için iyi bir ölçüdür. O problem parametreleri kodlamayla ölçülür ve uygunluk fonksiyonuna girdi olarak kullanılır. Yüksek ihtimalle uygun olan bu üyeler tekrar üreme, çaprazlama ve mutasyon operatörleriyle seçilirler. Bazı problemler için bireyin uygunluğu, bireyden elde edilen sonuç ile tahmin edilen sonuç arasındaki hatadan bulunabilir. Daha iyi bireylerde bu hata sıfıra yakın olur. Bu hata genellikle, girişin tekrar sunulacak kombinasyonlarının ortalaması veya toplamıyla hesaplanır. Beklenen ve üretilen değer arasındaki korelasyon etkeni, uygunluk değerini hesaplamak için kullanılabilir [31].

Amaç fonksiyonu her bir kromozomun durumunu değerlendirmek için mekanizmayı sağlayan ana bir kaynaktır. Bu GA ve sistem arasında önemli bir bağlantıdır. Fonksiyon girdi olarak kodu çözülmüş şekilde kromozom alır ve kromozomun performansına bir ölçü olarak bir objektif değer üretir. Bu diğer kromozomlar için de

yapıldıktan sonra bu değerler kullanılarak, uygun değerler uygunluk fonksiyonuyla hesaplanıp belli bir düzende planlanır. Bu planlamayı sağlayan ve uygunluk teknikleri olarak bilinen birçok yöntem vardır. Çoğu ortak kullanılan bu yöntemler şunlardır:

- **Genetik operatörler**

Pek çok GA operatörü öne sürülmüş olmasına rağmen en yaygın olarak kullanılanları seleksiyon, çaprazlama ve mutasyondur. Eski nesilden, yeni bir kromozom seti (nesil) üretebilmek için seleksiyon, çaprazlama ve mutasyon gibi genetik operatörler uygulanır.

Seleksiyon (Selection / Reproduction)

Yeniden üretme operatörü, hazır topluluktan uygun olan bireylerin seçilmesi ve bunların sonraki topluluğa kopyalanarak hayatta kalmalarıyla ilgilidir. Seçim modeli, tabiatın hayatta kalabilmek için uygunluk mekanizması modelidir. Yeniden üretme işleminde, bireyler onların uygunluk fonksiyonlarına göre kopya edilirler. Uygunluk fonksiyonu, mümkün olduğu kadar yükseltilmesi gereken bazı faydalı ve iyi ölçülerdir. Topluluk uzayındaki her bir bireyin uygunlukları baz alınarak ne kadar sayıda kopyasının olacağına karar verilir. En iyi bireylerden daha fazla kopya alınır, en kötü bireylerden kopya alınmaz. Bu, hayatta kalmak için uygunluk stratejisinin GA ya sağladığı avantajdır. Seçim aşağıdaki yöntemlerle olur.

Uyum orantılı seçim

Her seçilimde bir f_i bireyinin seçilme olasılığı bireyin mutlak uyum değerinin, tüm popülasyonun mutlak uyum değerine oranıdır. Fakat bu mekanizma hakkında bazı problemler vardır.

Diğerlerinden çok daha iyi olan bireyler, çok kısa bir sürede popülasyona hakim olmaktadır. Buna erken yakınsama denir.

Uyum değerleri birbirine çok yakın olduğunda neredeyse hiç seçilim baskısı olmamaktadır. Bu durumda biraz daha uyum değerinin yüksek olması çok büyük bir yarar sağlamamakta ve seçilim neredeyse düzgün rastsal bir seçilime dönüşmektedir. Burada sadece çok küçük bir farkla daha az uyuma sahip bireyler elenmekte ve çok yavaş bir evrim gerçekleşmektedir.

Mekanizma aynı uyum fonksiyonunun transpozelerinde farklı bir davranış içine girmektedir.

Sıralı seçim

Sıra tabanlı seçim uyum orantılı seçimin dezavantajlarına yoğunlaşılması üzerine bulunmuş bir diğer metottur. Doğrudan uyum değerlerini kullanarak bireylere seçilim olasılıkları atamak yerine, popülasyonu uyum tabanında sıralayarak ve bireylere sıralarına göre seçilim olasılıkları atayarak sürekli bir seçim baskısı oluşturur. Sıra numarası ile olasılıklar arasındaki haritalama pek çok farklı şekilde yapılabilir, örneğin lineer ya da eksponansiyel azalan olabilir. Tabi ki popülasyon toplamında olasılıklar toplamı bire eşit olmalıdır.

Rulet çarkı algoritması

Yukarıda popülasyondaki her birey için üremede seçilme olasılığını veren iki alternatif yöntem açıklanmıştır. İdeal bir dünyada üremede rol alma açısından eşleşme havuzundaki ebeveynlerin buradaki olasılık dağılımına eşit oranlar alması gerekir. Fakat bu pratikte popülasyonun sınırlı büyüklüğü nedeniyle olası değildir. Örneğin bireylerin beklenen kopya sayılarına bakarsak tamsayı olmayan değerler görürüz. Kısacası eşleşme havuzu olasılık dağılımından örneklenmektedir fakat onu tam olarak yansıtamamaktadır. Bu örneklemeyi en kolay yolla yapmanın yolu Rulet Çarkı Algoritması olarak bilinen yöntemdir. Bu konsept olarak deliklerin büyüklüğünün seçilim olasılıklarını belirttiği tek kollu bir rulet çarkını çevirmek gibidir. Eğer algoritmanın eşleşme havuzundaki m ebeveyn içinden l üyenin seçimi için kullanılacağını varsayarsak genelde aşağıdaki gibi uygulanır. Sıralama ile ya da rassal olarak 1’den m’ye kadar bir sıra varsayılır.

Turnuva seçimi

Bundan önceki metotlarda popülasyon üzerindeki bir bilgiye dayanarak bir olasılık dağılımından örnekleme yapmaya yaramaktaydı. Fakat bazı özel durumlarda, örneğin popülasyon çok büyük ise ya da popülasyon bir şekilde dağıtıksa, bu bilgiyi almak çok fazla zaman alabilir ya da en kötü ihtimalle imkansız olabilir. Bir başka durumda ise bir genel uyum fonksiyonu olmayabilir. Turnuva seçimi popülasyon hakkında global bir bilgiye ihtiyaç duymayan kullanışlı bir operatördür. Bunun yerine sadece iki bireyi karşılaştıracak bir sıralama bağıntısına dayanmaktadır. Dolayısıyla konsept olarak kurulumu ve uygulanması daha kolay ve hızlıdır. Turnuva seçimi, sıralama yöntemi gibi mutlak uyuma değil sadece bağıl uyuma baktığından uyum fonksiyonunun transpozesinin alınması durumunda sonuçlar değişecektir. Turnuva seçiminde bir bireyin seçilmesi aşağıdaki dört faktöre dayanır:

Popülasyondaki sırasına. Fakat bu tüm popülasyonu sıralamadan tahmin edilir.

Turnuvaya dahil edilen birey sayısı arttıkça turnuvanın sonucunda daha büyük uyuma sahip bireylerin çıkma ihtimali artar, daha az uyuma sahip bireylerin çıkma ihtimali azalır.

Turnuvanın en uygun bireyinin seçilme ihtimali olan p olasılığı. Bu genelde deterministik sistemlerde $p=1$ olarak alınır fakat stokastik sistemlerde $p<1$ olarak kullanılır. İkinci durumda seçim baskısı doğal olarak daha azdır.

Eğer deterministik ve yer değiştirmeli bir turnuva yapılıyorsa en az uyumlu bireyin seçilme ihtimali yoktur. Fakat yer değiştirmeli bir seçim yapılıyorsa en az uyumlu bireyin bile şanslı bir çekiliş sonrasında seçilebilme ihtimali her zaman vardı.

Çaprazlama

Seçilen bireylerin çiftleşerek popülasyona yeni olası problem çözümleri katması bakımında oldukça önemlidir. Problem türü doğrultusunda, seçilen farklı kodlama sistemleriyle uyumlu olacak çaprazlama operatörleri geliştirilmiş. Öncelikle ikili

kodlama sisteminde kullanılan çaprazlama operatörleri ardından reel kodlamada kullanılan operatörlerden bahsedilecektir

İkili Kodlamada Çaprazlama

İkili kod çaprazlama yöntemlerinin açıklanmasında kolaylık sağlaması açısından örnek A ve B kromozomları A:10010010010 ve B:11000110101 olarak seçilmiştir

a) **Tek Noktalı Çaprazlama:** Çaprazlama için rasgele bir nokta belirlenir. Çaprazlamaya giren iki birey bu noktadan koparılır ve kopma noktasından sonraki genler yer değiştirilerek 2 yeni çocuk elde edilir. Rasgele seçilen noktanın 3 olduğu varsayılırsa çaprazlama sonrası oluşan aşağıdaki çocuklar oluşacaktır.

Ç1: 10000110101
Ç2: 11010010010

b) **Çok Noktalı Çaprazlama:** Çaprazlama için x adet rasgele nokta belirlenir. Çaprazlamaya giren bireylerin bu noktalar arasında kalan genleri yer değiştirilerek 2 yeni çocuk elde edilir. Bu yöntem çeşitliliği daha çok artırarak arama alanını genişletir ve başlangıçta en iyi çözüm gibi gözüken bir noktaya yakınsama olasılığını azaltır. Rasgele seçilen x=2 noktanın 3 ve 7 olduğu varsayılırsa çaprazlama sonrası aşağıdaki çocuklar oluşacaktır.

Ç1: 10000110010
Ç2: 11010010101

c) **Düzgün Çaprazlama:** Bu çaprazlamada 0 ve 1 değerlerinin rasgele atanmasıyla oluşturulan birey uzunluğunda bir maskeler kullanılır. Maskeler çocukların hangi genlerinin hangi ebeveynlerden geleceğini tayin eder. Rasgele atanan maskeler birinci ve ikinci çocuk için sırasıyla 12121122122 ve 11211221212 ise çaprazlama sonrası aşağıdaki çocuklar oluşacaktır.

Ç1: 11000010001 Ç2: 10010110111

Mutasyon (Mutation)

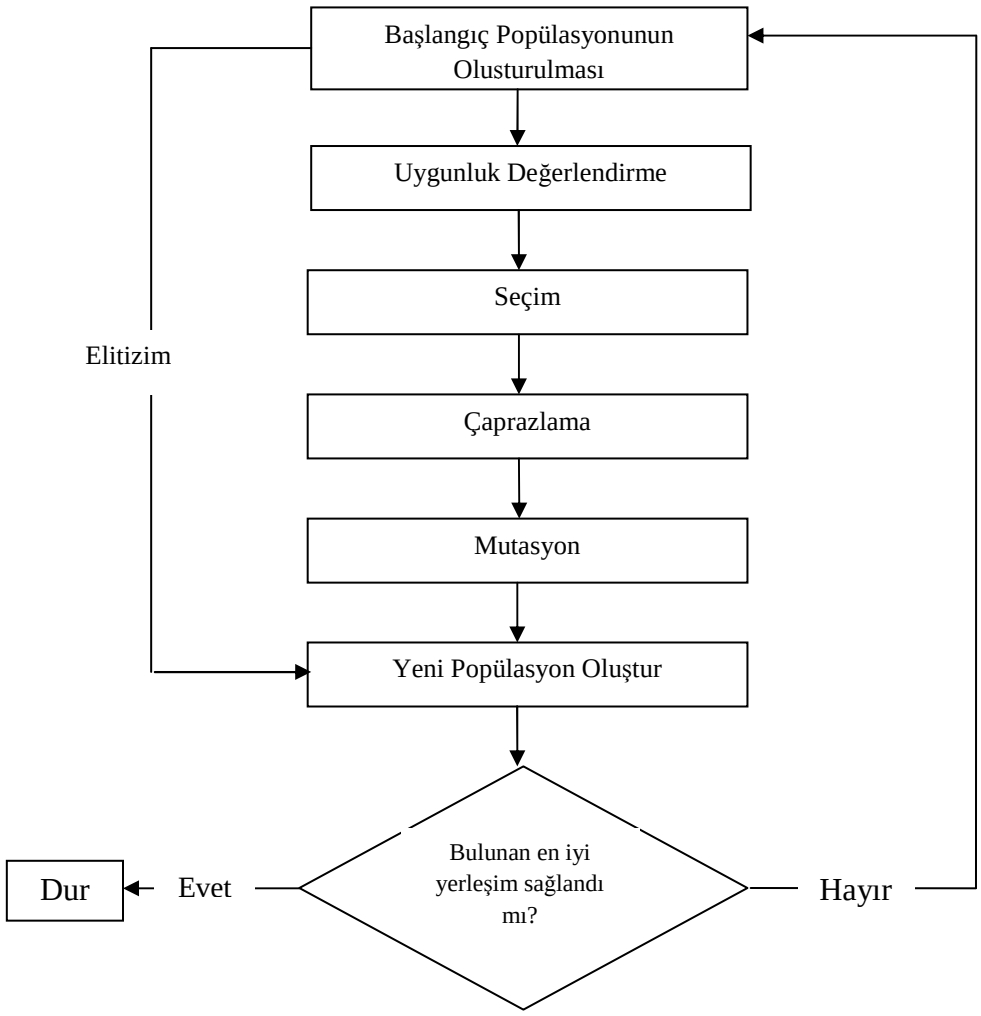
Amaç, var olan bir kromozomun genlerinin bir ya da birkaçının yerlerini değiştirerek yeni kromozom oluşturmaktır. Yeniden ve sürekli yeni nesil üretimi sonucunda belirli bir süre sonra nesildeki kromozomlar birbirlerini tekrarlama konumuna gelebilir ve bunun sonucunda farklı kromozom üretimi durabilir veya çok azalabilir. İşte bu nedenle mutasyon bazı yeni oluşturulan çocuklarda zoraki olarak gerçekleştirilir. Böylece çözümlerin lokal optimumlara yakınsaması engellenir [34, 36].

Açıklandığı gibi mutasyonun birinci maksadı bir popülasyonun içindeki değişimi tanımlamaktır. Mutasyon popülasyonlarda çok önemlidir. Öyle ki burada ilk popülasyon mümkün olan tüm alt çözümlerin küçük bir alt kümesi olabilir ve ilk popülasyondaki tüm kromozomların önemli biti sıfır olabilir. Halbuki o bitin problemin çözümü için 1 olması gerekebilir ve bunu da çaprazlama düzeltemeyebilir. Bu durumda o bit için mutasyon kaçınılmazdır. Genellikle önerilen mutasyon oranı 0.005/bit/jenerasyondur. Bu ilsem çaprazlamadan sonra gelir. Mutasyonun yapılıp yapılmayacağını bir olasılık testi belirler. Örneğin yeni neslin ortalama uygunluğu ≤ Eski neslin ortalama uygunluğu ise; x. Kromozomun y. Bitini değiştir denilebilir.

Seçkincilik (Elitism)

Bu şema, şu an popülasyondaki en uyumlu üyeyi kaybetmeyi önlemek amacıyla genellikle yaş-tabanlı ve stokastik uyum-tabanlı yer değiştirme şemaları ile birlikte kullanılır. Gerçekte, şu an popülasyondaki en uyumlu üyenin izi vardır ve bu daima popülasyonda saklanacaktır. Bu nedenle, bu, eğer grup içinde yer değiştirmek üzere seçilirse ve popülasyona sokulmakta olan hiçbir çocuk eşit veya daha iyi uyuma sahip değilse, o zaman bu saklanır ve çocukların biri atılır.

3.3.3. Genetik algoritmaların akış diyagramı



Şekil 3.12. GA Akış Diyagramı

3.3.4. Genetik algoritmalarda parametre seçimi

Parametreler, genetik algoritma performansı üzerinde önemli etkiye sahiptir. Optimal kontrol parametreleri bulmak için bir çok çalışma yapılmıştır fakat tüm problemler için genel olarak kullanılabilecek parametreler bulunamamıştır. Bu parametreler, kontrol parametreleri olarak adlandırılmaktadır. Kontrol parametreleri populasyon büyüklüğü, çaprazlama olasılığı, mutasyon olasılığı, kuşak aralığı, seçim stratejisi ve fonksiyon ölçeklemesi olarak sayılabilir. Bu parametreler aşağıda açıklanmıştır

Populasyon büyüklüğü: Genetik algoritma kullanıcısı tarafından verilen en önemli kararlardan birisidir. Bu değer çok küçük olduğunda, genetik algoritma yerel bir optimuma takılabilmektedir. Populasyonun çok büyük olması ise çözüme ulaşma zamanını arttırmaktadır. Bu konuda Goldberg 1985'de, yalnızca kromozom uzunluğuna bağlı bir populasyon büyüklüğü hesaplama yöntemi önermiştir. Ayrıca Schaffer ve arkadaşları 1989'da çok sayıda test fonksiyonları üzerinde yaptıkları araştırmalar sonucunda, 20-30 arası bir populasyon büyüklüğünün iyi sonuçlar verdiğini belirtmişlerdir [36].

Çaprazlama olasılığı: Çaprazlamanın amacı, mevcut iyi kromozomların özelliklerini birleştirerek daha uygun kromozomlar yaratmaktır. Kromozom çiftleri P(c) olasılığı ile çaprazlamaya uğramak üzere seçilirler. Çaprazlamanın artması, yapı bloklarının artmasına neden olmakta fakat aynı zamanda bazı iyi kromozomların da bozulma olasılığını arttırmaktadır [36].

Mutasyon olasılığı: Mutasyonun amacı populasyondaki genetik çeşitliliği korumaktır. Mutasyon P(m) olasılığı ile bir kromozomdaki her bitte meydana gelebilir. Eğer mutasyon olasılığı artarsa, genetik arama rastsal bir aramaya dönüşür. Bu aynı zamanda kayıp genetik malzemeyi tekrar bulmada yardımcı olmaktadır.

Kuşak aralığı: Her kuşaktaki yeni kromozom oranına kuşak aralığı denmektedir. Genetik operatörler için kaç tane kromozomun seçildiğini gösterir. Yüksek bir değer birçok kromozomun yer değiştirdiği anlamına gelmektedir [35-36].

Seçim stratejisi: Eski kuşağı yenilemenin çeşitli yöntemleri mevcuttur. Kuşaksal stratejide, mevcut populasyondaki kromozomlar tamamen yavrular ile yer değiştirir. Populasyonun en iyi kromozomu da yenilendiğinden dolayı bir sonraki kuşağa aktarılamaz ve bu yüzden bu strateji en uygun (elitist) stratejisiyle beraber kullanılmaktadır. En uygun stratejisinde, populasyondaki en iyi kromozomlar hiçbir zaman yenilenmemektedir, bundan dolayı çoğalma için en iyi çözüm her zaman elverişlidir. Denge durumu stratejisinde ise, her kuşakta yalnızca birkaç kromozom yenilenmektedir. Genellikle, yeni kromozomlar populasyona katıldığında en kötü kromozomlar yenilenir [36].

Fonksiyon ölçeklemesi: Doğrusal ölçekleme, üstsel ölçekleme gibi yöntemler mevcuttur. Probleme göre en uygun ölçekleme yönteminin seçilmesi genetik algoritmanın etkin işlemesi açısından önem taşımaktadır [36].

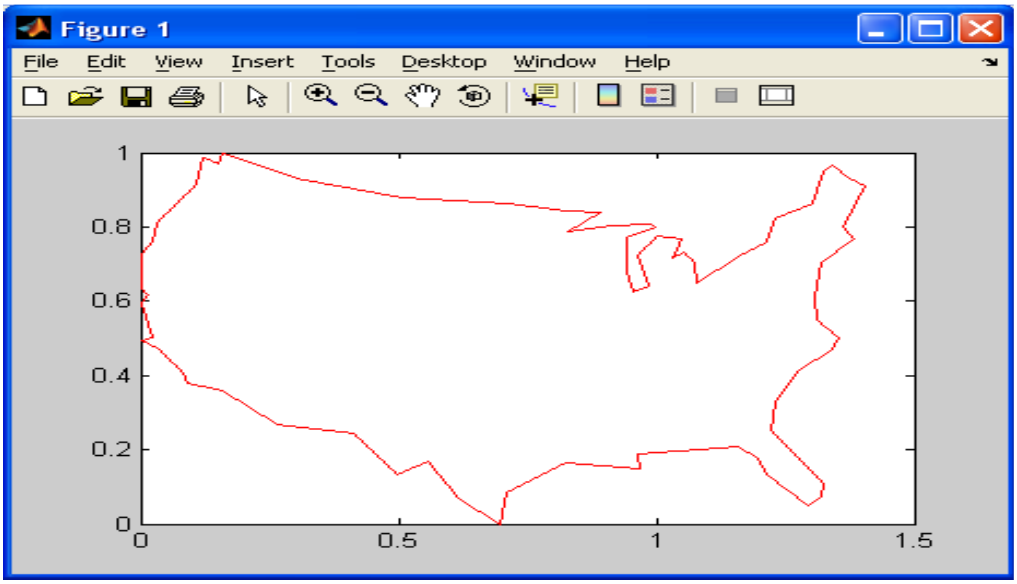
Örnek

Gezgin satıcı probleminin örneği tam sayılı optimizasyonla çözümü gösterilmişti. Bu problem tipi GA ile aşağıdaki adımları takip ederek çözülür [38].

1.adım: Gezgin satıcının gezeceği güzergahı oluşturmak için MATLAB içerisinde USBORDER.mat adlı dosya bulunmaktadır. Bu dosyayı aktif etmek için

“load('usborder.mat','x','y','xx','yy')
plot(x,y,'Color','red'); hold on”

komutlarını MATLAB’da bulunan komut penceresinde yazmak gerekmektedir, örnek olarak şekil 3.13. de görülen harita ele alınmıştır.



Şekil 3.13. Gezgin Satıcı İçin Ülke Haritası

Bu aşamadan sonra harita içerisine satıcının gezeceği şehirleri eklemek gerekmektedir. Bunu için;

“cities = 9

locations = zeros(cities,2)

n = 1

while (n <= cities)

xp = rand*1.5

yp = rand

if inpolygon(xp,yp,xx,yy)

locations(n,1) = xp

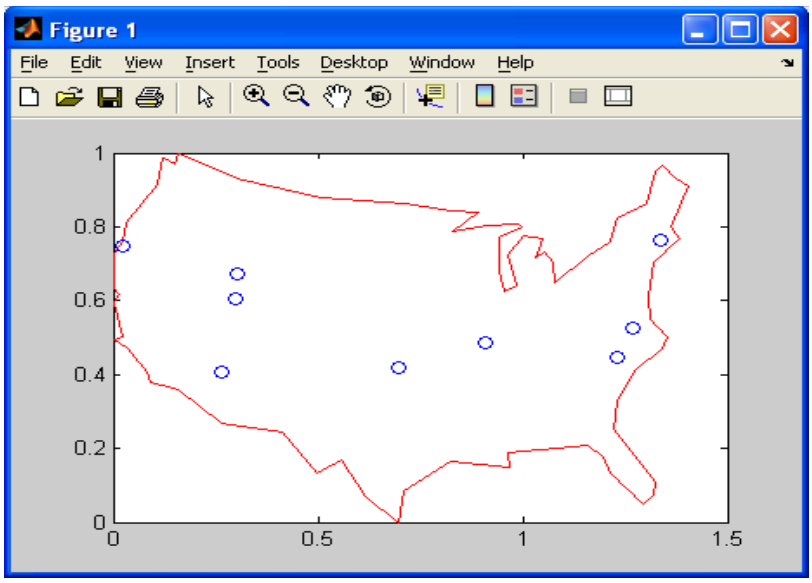
locations(n,2) = yp

n = n+1

end

end

plot(locations(:,1),locations(:,2),'bo')” komutlarını yazarak aşağıda şekil 3.14 elde edilir.



Şekil 3.14. Rassal Olarak Atanan Şehirler

Şehirler oluşturduktan sonra bu şehirler arasındaki mesafeyi elde etmek gerekmektedir. Bunu için ise;

“distances = zeros(cities)

for count1=1:cities

for count2=1:count1

x1 = locations(count1,1)

y1 = locations(count1,2)

x2 = locations(count2,1)

y2 = locations(count2,2)

distances(count1,count2)=sqrt((x1-x2)^2+(y1-y2)^2)

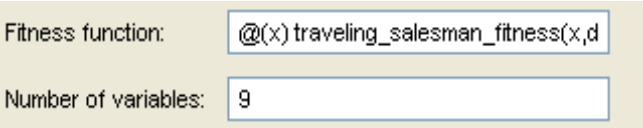
distances(count2,count1)=distances(count1,count2)

end

end”

satırlarını MATLAB komut bölmesine yazmak gerekmektedir. Bu komutları yazıldıktan sonra MABLAB Genetik Algoritmalar eklentisi çalıştırılmalıdır. GA çalıştırıldığı zaman;

Şekil 3.15. te Fitness Function = @(x) traveling_salesman_fitness(x,distances)
Number of Variables = 9 olarak belirlenir.



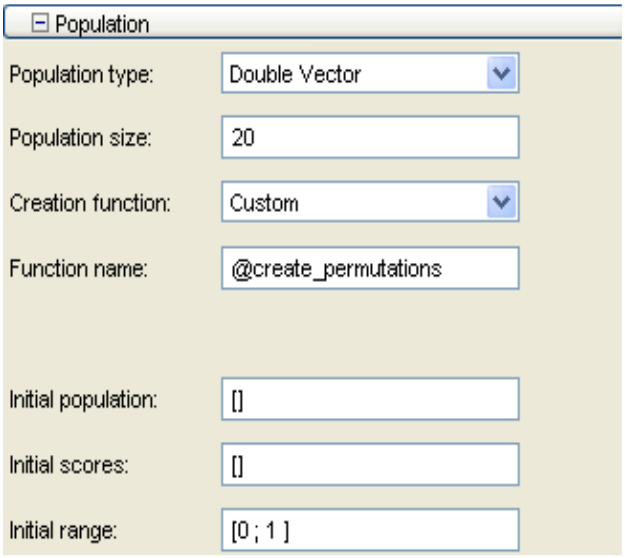
Şekil 3.15. Uygunluk Fonksiyonu

Şekil 3.16. da Popülasyon ile ilgili özellikler olara aşağıda belirtilen işlemler yapılacaktır.

Population type: double vector

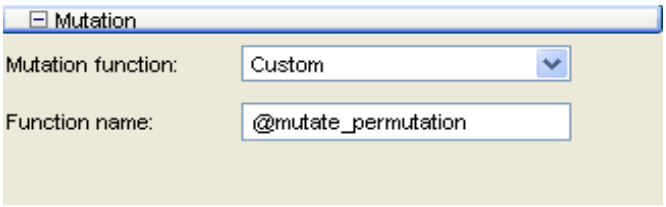
Population size: 20

Creation function = custom özelliği seçilerek açılan pencereye @create_permutations yazılacaktır.



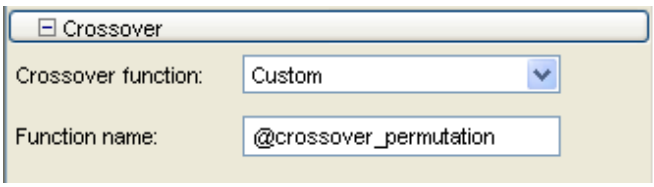
Şekil 3.16. Populasyon

Şekil 3.17. de Mutation (mutasyon) ile ilgili özellikler olara aşağıda belirtilen işlemler yapılacaktır. Mutation function olarak custom seçeneği seçilerek açılan kısıma @ mutate_permutation yazılacaktır.



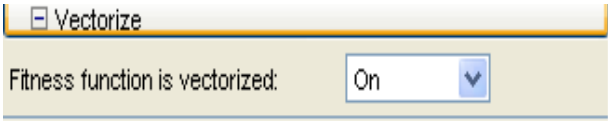
Şekil 3.17. Mutasyon

Şekil 3.18. de Crossover (çaprazlama) ile ilgili özellikler olarak aşağıda belirtilen işlemler yapılacaktır. Crossover function olarak custom seçeneği seçilerek açılan kısıma @ crossover_permutation yazılacaktır.



Şekil 3.18. Çaprazlama

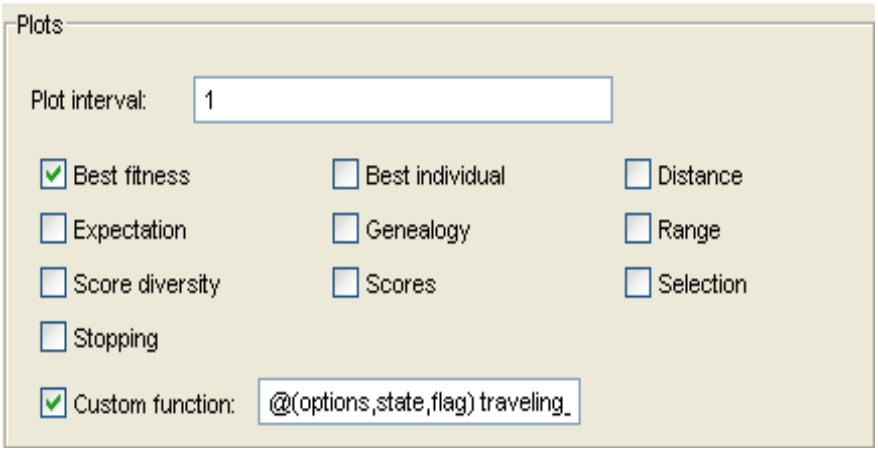
Şekil 3.19. da Vectorize özelliği ile ilgili olarak aşağıda belirtilen işlemler yapılacaktır.



Şekil 3.19. Vektörleştirme

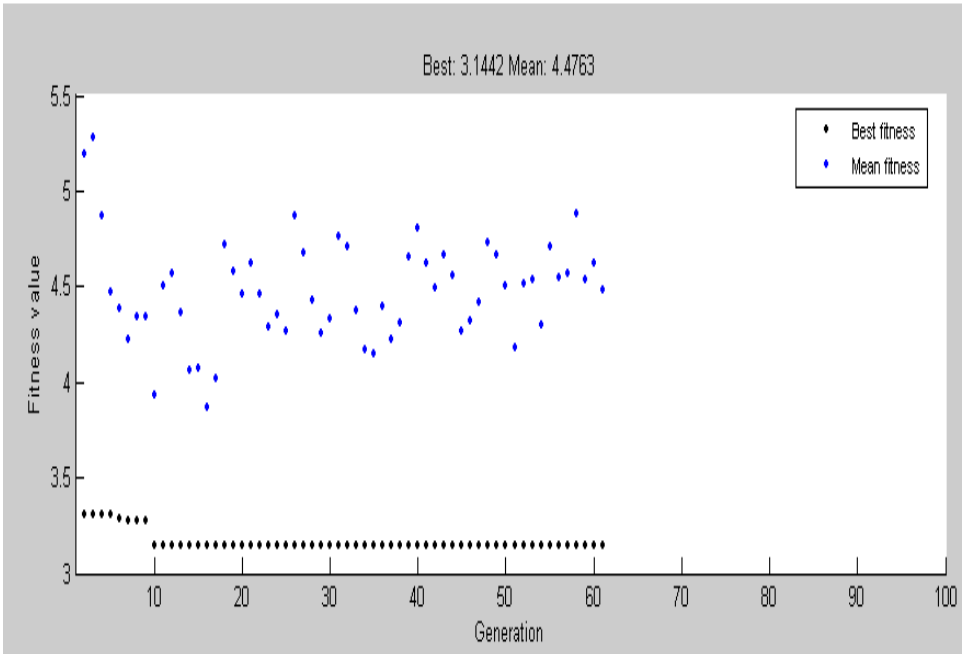
3.3.5. ÇİZİMLER

Görmek istenilen raporların yanı sıra aşağıdaki şekil 3.20 de ki gösterilen Custom function kısmına;
@(options,state,flag) traveling_salesman_plot (options,state,flag,locations) yazılır.
Böylece ilk başta oluşturulan harita ve şehirler arasındaki değişimi takip edilebilir.



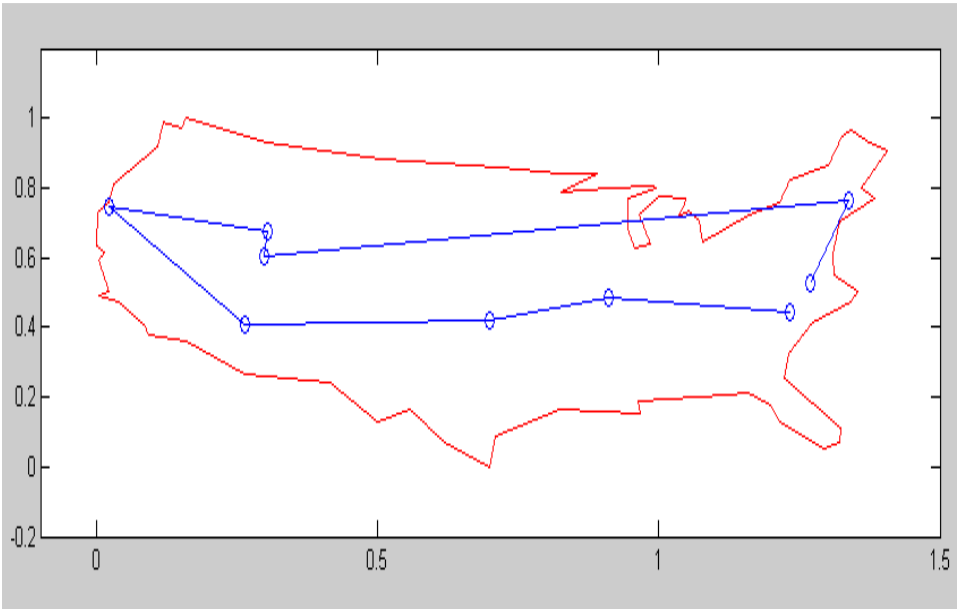
Şekil 3.20 Çizim Seçimi

Bu işlemleri tamamladıktan sonra GA araç kutusunu çalıştırıp, sonuçlar elde edilir.



Şekil 3.21. Jenerasyon ve Uygunluk Değerleri

Burada en iyi değerin 3,1442 olduğu ortalamanın ise 4,4763 olduğunu görülüp güzegahın aşağıda Şekil 3.22 de ki gibi olması gerektiği görülüyor.

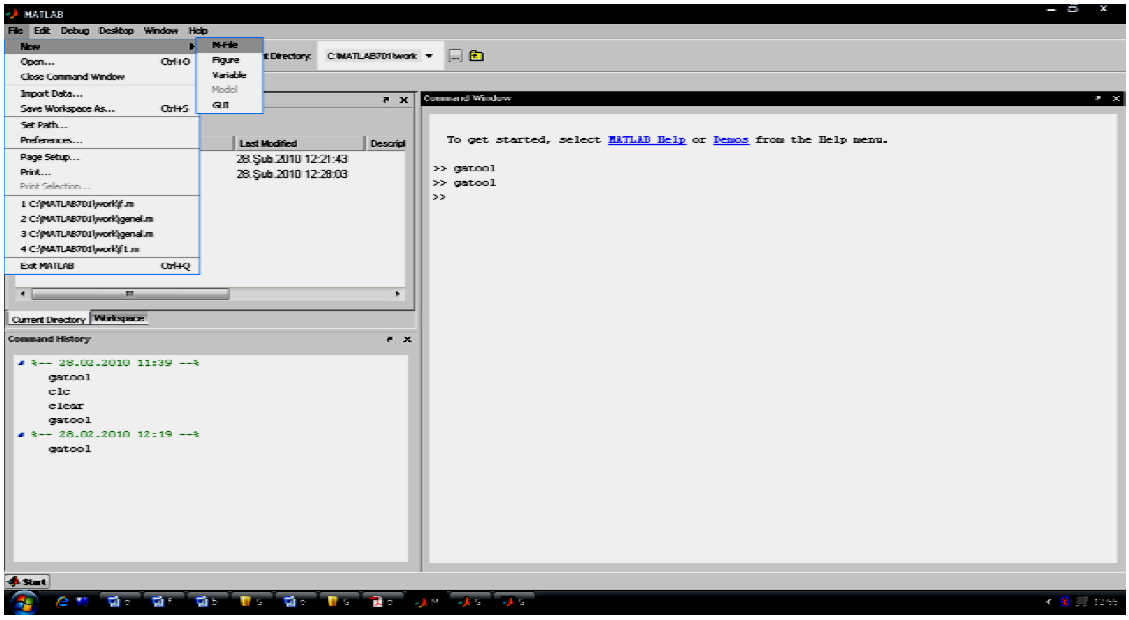


Şekil 3.22 Güzergahlar

Örnek

fonksiyonunu BFGS metodlu Newton-Benzeri algoritmasıyla minimumlaştırılması Bölüm 1 de gösterilmişti, (Bkz. 2.Bölüm sayfa 72) şimdi bu fonksiyonun minimizasyonu, GA kullanılarak gösterilecektir.

Adım1: MATLAB M file oluşturulması

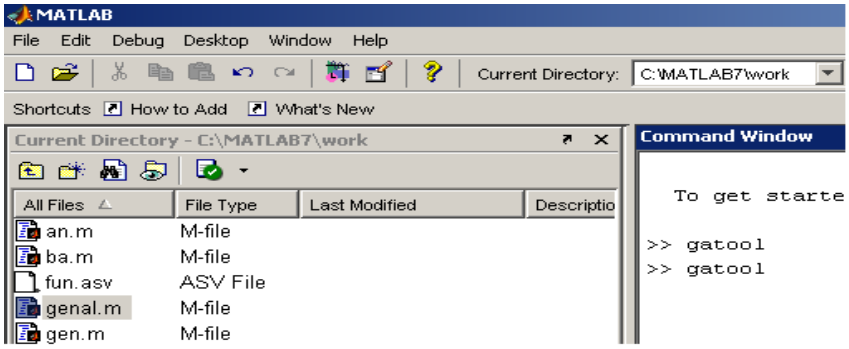


Şekil 3.23 M File Oluşturma

```
function z=f(x)
z=[5*(x(1)^2)+2*(x(2)^2)+2*(x(3)^2)+2*x(1)*x(2)+2*x(2)*x(3)-2*x(1)*x(3)-6*x(3)];
range = [-10,10;-10,10;-10,10];
```

şeklinde yazılır.

Adım 2: Command window ekranı üzerinde GA araç kutusu çağrılmalı.

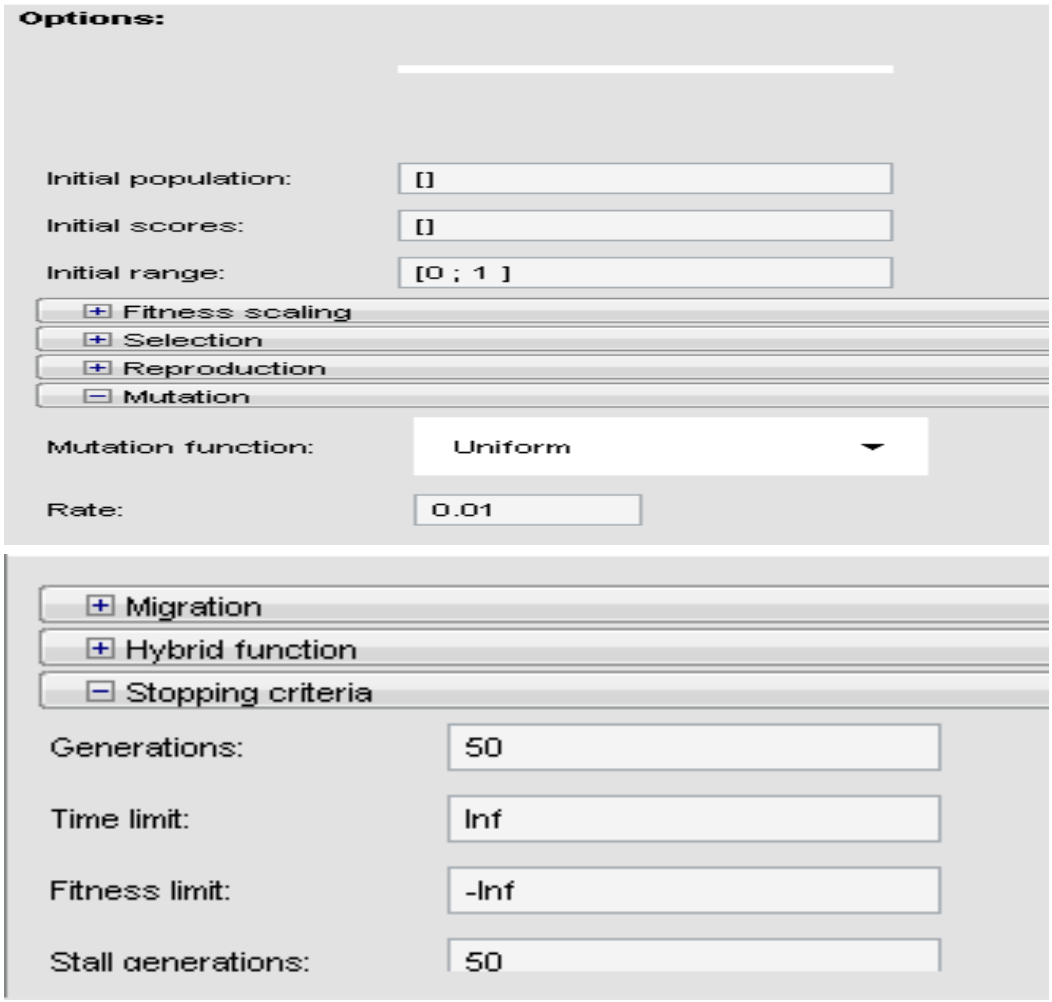


Şekil 3.24. Command Window Ekranı

Adım 3: Şekil 3.25 te gösterilen gelen GA tool ekranına @fonksiyon adı şeklinde fonksiyonun adı yazılır. Fonksiyonun x_1, x_2, x_3 olan üç değişkeni M file üzerindeki hali ile x(1) ve x(2) olmak üzere 3 adet değişken içermektedir. Bu nedenle GA Tool üzerinde gösterilen Number of Variables kısmına 3 yazılmalı.

Şekil 3.25. Uygunluk Fonksiyonu ve Değişkenler

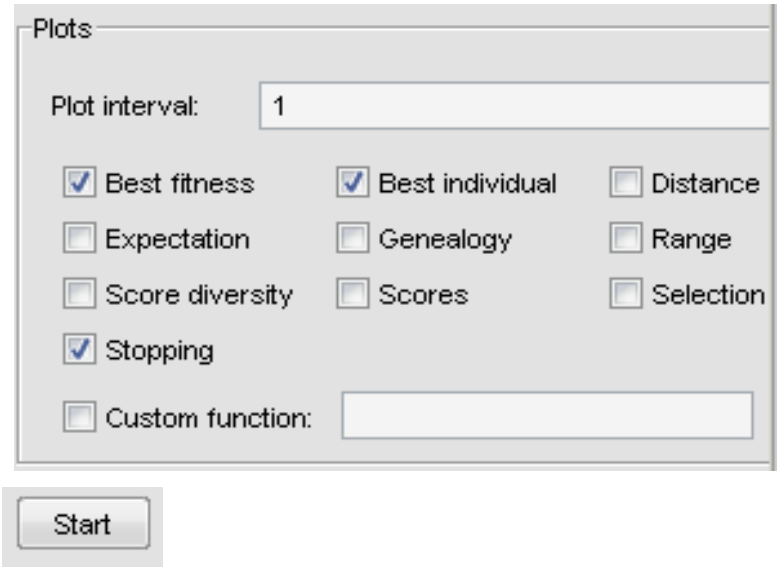
Daha sonra GA nın parametrelerinin girişi ayarlar (options) kısmından aşağıda şekil 3.26. da gösterildiği gibi yapılmalıdır.



Şekil 3.26. Ayarlar

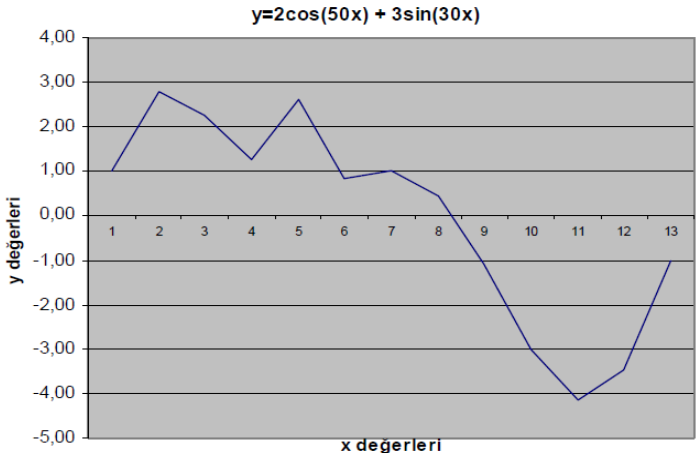
Populasyon büyüklüğünü yukarıda açıklandığı gibi 50 ve [0,1] aralığında belirlenir. Çaprazlama yöntemi olarak tek noktada çaprazlama (single point) ve mutasyon oranı olarak uniform, oranı 0,01 olarak belirtilir. Daha sonra istenilen generasyon sayısı yazılmalı, bu problem için 50 jenerasyon oluşturmak istenildiğinden, stopping criteria kısmına 50 yazılır.

Adım 4: GA çalıştırma aşağıda şekil 3.27 de gösterildiği gibi yapılır.



Şekil 3.27. GA Çalıştırma

Örnek: Burada, genetik algoritmaların nasıl çalıştığını göstermek için tek değişkenli bir fonksiyonun optimizasyonu ele alınacaktır. Örnek fonksiyon aşağıdaki gibi olup, grafiği şekil 3.28 de gösterilmiştir:



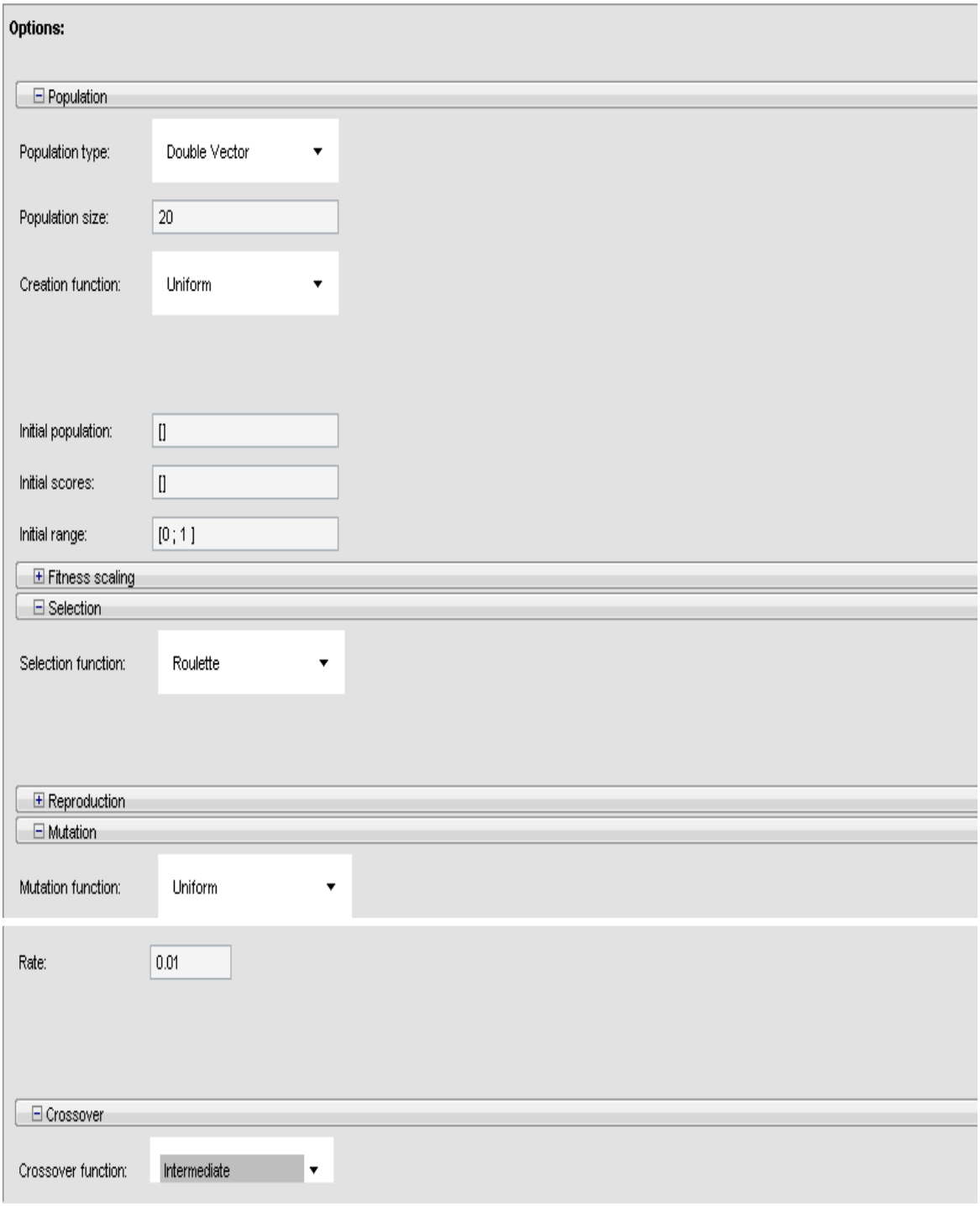
$y = 2\cos(50x) + 3\sin(30x)$

$0 \leq x \leq 12$ olmak üzere fonksiyonun grafiği yandaki gibidir.

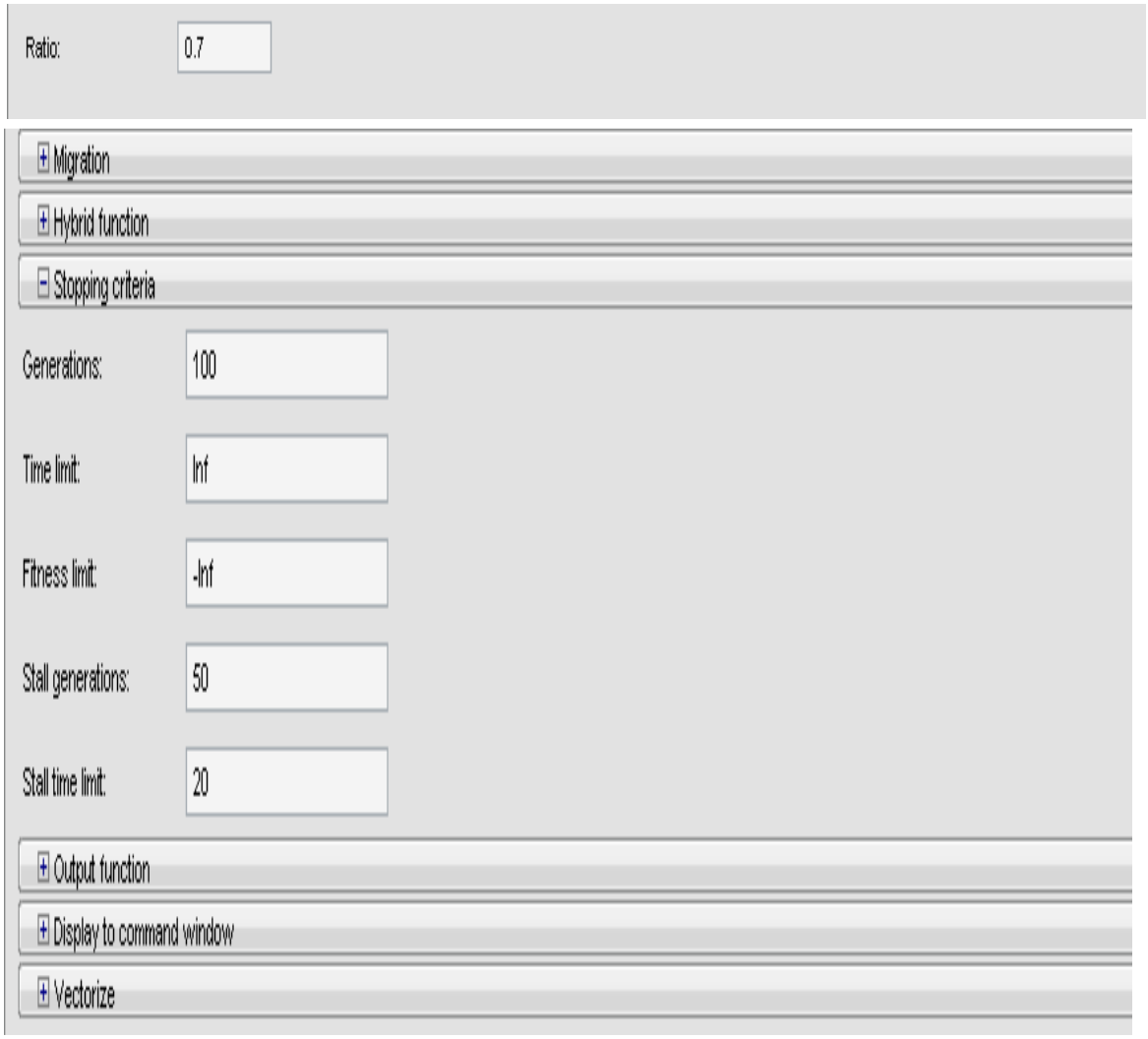
Şekil 3.28. y Fonksiyonunun Grafiği

Burada sadece genetik algoritmayla yapılan çözümün sonucu ile karşılaştırma amacıyla hesaplanmıştır. Genetik algoritmayla fonksiyonun optimum değerini bulmak için Goldberg'in basit genetik algoritma kodu örneğe uyarlanarak kullanılmıştır. Problemin çözümü için ikili kodlama kullanılmış ve kromozomlar 30

bitlik dizilerden oluşturulmuştur. Kromozom uzunluğu genellikle gerçek uygulamalarda uzun alınmaktadır. Uygulama Şekil 3.29. da gösterilmiştir.



Şekil 3.29. Matlab GA Araç Kutusunda Uygulama



Şekil 3.29' a devam

Yukarıda bir önceki örnekte anlatıldığı gibi GA nın MATLAB araç kutusunda bu problem için olan veriler gösterilmiştir. Program çalıştırıldığında fonksiyonun minimum değeri -4,5597 ve fonksiyonu minimum yapan X değeri ise 11,17758 olarak bulunur.

*Aynı fonksiyon klasik yöntemle çözümlenseydi;

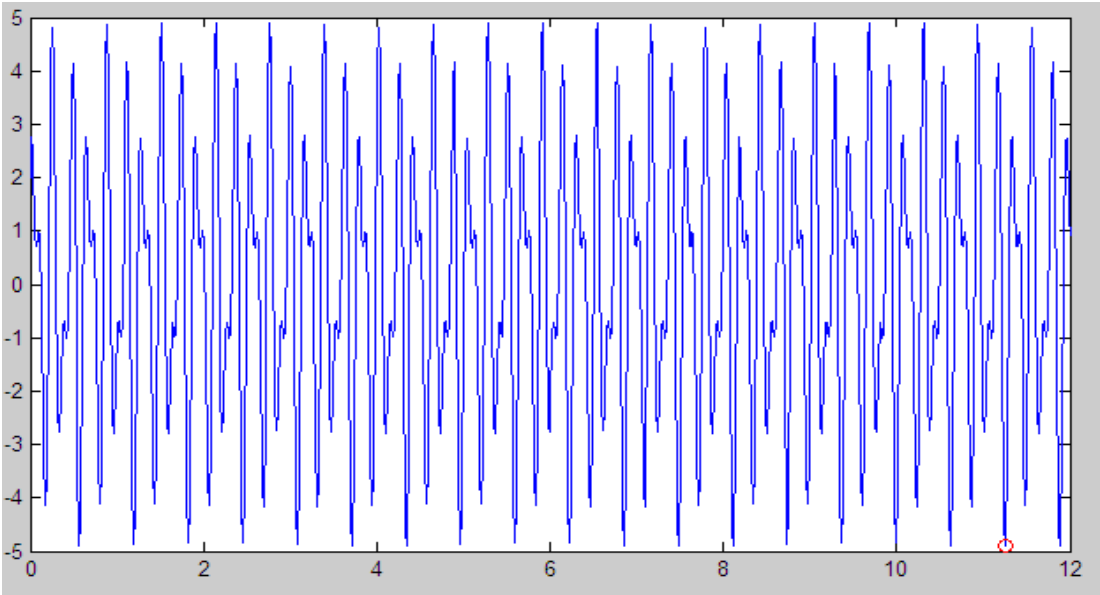
y = 2cos(50x) + 3sin(30x)

$0 \leq x \leq 12$

Verilen fonksiyon türevlenebilir ve diferansiyel matematik kuralları göz önüne alınarak maksimum yada minimum değeri klasik yöntemlerle araştırılabilir.

```
function minimumf
format long
x=0:0.01:12;
z=f(x);
a=min(z);
min1=0
for i=1:length(z)
if z(i)==a
minimum=x(i);
min1=[min1;minimum];
end
end
sonuc=min1(2:length(min1));
disp('mnokta  değeri')
disp([sonuc,a])
plot(x,z)
hold on
for i=1:length(sonuc)
plot(sonuc(i),a,'ro')
end
function y=f(x)
y=2*cos(50*x)+3*sin(30*x);
```

şeklinde Matlab programında yazılacak kod çözüm için yeterlidir. Çözüm aşağıdaki gibi olur.



Şekil 3.30 Klasik Metod Çözümü İle Ekran Çıktısı

Minimum Nokta Değeri Noktanın Değeri
11.250000000000000 -4.902947156437075

Yukarıdaki örneklerdeki analizlerden görüldüğü gibi GA etkin çözümlemelere sahip bir zeki optimizasyon tekniğidir. GA da altı çizilmesi gereken önemli bir nokta programın MATLAB ortamında çözümünün gerçekleştirilirken algoritmanın her çalıştırıldığında rassal sayılar nedeniyle farklılıklar göstermesinden dolayı programın bir kez değil birçok defa çalıştırılarak sonuçların elde edilmesi gerektiğidir.

3.4. Bulanık Optimizasyon

“Bilgisayarlar insan beyni gibi çalışmazlar. İnsan beyni, “su sıcak”, “hız yüksek”, “adam genç” gibi objektif, tam ve kesin olmayan yargılar ile çalışabilirken, bilgisayarlar bir konuda karar verebilmek için doğru ya da yanlış olduğu birler ve sıfırlar ile ifade edilebilecek kadar indirgenmiş verileri kullanırlar.” [39] Bu durum bilim dünyasında bilgisayarların insanlar gibi düşünüp düşünemeyecekleri tartışmalarının çıkmasına neden olmuştur. Bu tartışmalar Prof Dr. Lotfi Zadeh’in Bulanık Mantık kavramını ortaya atmasına kadar sürmüştür.

“Bulanık mantık ile klasik mantık arasındaki temel fark bilinen anlamda matematiğin sadece aşırı uç değerlerine izin vermesidir. Klasik matematiksel yöntemlerle karmaşık sistemleri modellemek ve kontrol etmek işte bu yüzden zordur, çünkü veriler tam olmalıdır. Bulanık mantık kişiyi bu zorunluluktan kurtarır ve daha niteliksel bir tanımlama olanağı sağlar. Bir kişi için 35,5 yaşında demektense sadece orta yaşlı demek birçok uygulama için yeterli bir veridir. Böylece azımsanamayacak ölçüde bir bilgi indirgenmesi söz konusu olacak ve matematiksel bir tanımlama yerine daha kolay anlaşılabilen niteliksel bir tanımlama yapılabilecektir ” [39].

Bulanık mantıkta bulanık kümeler kadar önemli bir diğer kavram da dilsel (linguistik) değişken kavramıdır. Dilsel değişken “yakın” veya “uzak” gibi kelimeler ve ifadelerle tanımlanabilen değişkenlerdir. Bir dilsel değişkenin değerleri bulanık kümeler ile ifade edilir. Örneğin oda sıcaklığı dilsel değişkeni için “sıcak”, “soğuk” ve “çok sıcak” ifadelerini alabilir. Bu üç ifadenin her biri ayrı ayrı bulanık kümeler ile modellenir.

Bulanık mantığın ana kavramı bulanık kümelerdir. Buradaki küme kavramının anlaşılması oldukça kolaydır. Örneğin “su ılık” ifadesindeki “ılık” kavramının sınırları kişiden kişiye değişiklik göstermekle birlikte, bu sınırlarda bir kesinlik söz konusu değildir. Fakat genel olarak 18 derece ile 25 derece arası ılık’lık kavramının sınırları olarak düşünülebilir. Bulanık mantığın sistemi Şu şekildedir. Bir ifade tamamen yanlış ise klasik mantıkta olduğu gibi 0 değerindedir, eğer tamamen doğru

ise 1 değerindedir. Bunların dışında tüm ifadeler 0'dan büyük 1 den küçük reel değerler alırlar

Hesaplamaların böyle yapılabileceği ilk defa 1965 yılında Azeri kökenli ABD' li bilim adamı Lotfi Ali Asker-Zadeh tarafından yayınlanmıştır. "Bulanık Kümeler" olarak adlanan bu makalede, matematiğin, dil ve insan zekasını ilişkilendirebileceği gösterilmiş ve bunun için bulanık kümeler teorisini teklif etmiştir. Zadeh birçok kavramın dilsel olarak geleneksel matematiğe göre daha iyi belirlenebildiğini ve bulanık mantığın ve onun bulanık kümelerdeki ifadelerinin gerçek hayatın daha iyi modelini oluşturduğunu göstermiştir.

Bulanık mantığın uygulama alanları çok geniştir. Sağladığı en büyük fayda ise insana "özgü tecrübe ile öğrenme" olayının kolayca modellenebilmesi ve belirsiz kavramların bile matematiksel olarak ifade edilebilmesine olanak tanımasıdır. Bu nedenle lineer olmayan sistemlere yaklaşım yapabilmek için özellikle uygundur. "Bulanık mantık konusunda yapılan araştırmalar Japonya'da oldukça fazladır. Özellikle "fuzzy process controller" olarak isimlendirilen özel amaçlı bulanık mantık mikroişlemci çipinin üretilmesine çalışılmaktadır. Bu teknoloji fotoğraf makineleri, çamaşır makineleri, klimalar ve otomatik iletim hatları gibi uygulamalarda kullanılmaktadır [39]. Bundan başka uzay araştırmaları ve havacılık endüstrisinde de kullanılmaktadır.

Bulanık mantık kuramının ilk önemli endüstriyel uygulaması çimento sanayisinde olmuştur. Bu sanayide değirmen içerisindeki sıcaklık ve oksijen oranı ürün kalitesi açısından çok önemlidir. Kısıtlı ve hassas olmayan, ısı ve karbonmonoksit oranı gibi bilgilerle iyi bir çalışma düzeni elde edilmesi bir sanat olup operatörlerin bu konuda yeterli bir uzmanlık kazanabilmeleri için inanılmaz farklılıklar olacağından, üretilen çimento da vardiyadan vardiyaya değişecek, tutarlı kalitede çimento üretimi çok zor olacaktır.

Bir Danimarka firması bu nedenlerden dolayı lineer bir model üzerine kurulu geleneksel denetleyici yerine bir bulanık mantık denetleyici kullanmayı düşünmüş ve çok başarılı sonuçlar veren bir uzman sistemi geliştirmiştir. Bu veya benzeri sistemler bugün bile Japonya ve Amerika da dahil olmak üzere birçok ülkede kullanılmaktadır. Kronolojik sıra içerisinde bundan sonraki en önemli aşama Japonya da 1987 yılında görülmüştür. Hitachi firması Sedai metro sisteminde çalışan trenlerin otomatik olarak denetimi için bulanık mantık kullanmıştır.

Geliştirilen sistemde, daha önce tren operatörü tarafından yapılan ve yolcuların sarsıntılı bir yolculuk geçirmelerine neden olabilen hızlanma raportörünün yapması gereken işler kapıları kapatmak ve başlatma düğmesine basmak gibi birkaç işlemle sınırlı kalmaktadır. Bu başarılı uygulamadan sonra bulanık denetim konusundaki çalışmalar yeni bir ivme kazanmış ve endüstriyel uygulama alanları hızla artmıştır. Çalışmaların uluslararası alanda koordinasyonu amacı ile Japonya'da 1989 yılında, LIFE adlı bir laboratuar kurulmuştur. Bulanık mantık denetleyiciler konusundaki kuramsal çalışmaları hala sürüyor olmasına rağmen artık bu konu endüstride kendisine önemli bir yer edinmiş durumdadır. Uygulama alanları arasında beyaz eşya, tren, asansör, trafik kontrolü ve otomotiv sanayi sayılabilir. Bugün Japonya'da bulanık denetim kullanan beyaz eşyalar ve elektronik aletler, örneğin fotoğraf ve çamaşır makineleri, güncel yaşamın birer parçasıdırlar.

Günümüzde birçok ülkede bulanık mantık konusunda araştırmalar yapmakta olup bunlar arasında ABD, Japonya, Çin ve Batı Avrupa ülkeleri başta gelmektedir. NASA bünyesinde bulanık denetim konusunda çalışan çok kuvvetli bir grup vardır. Bu grup uzay mekiği için pilotların yükünü azaltmak, sistemin güvenilirliğini artırmak ve yakıt tüketimini azaltmak amacıyla bir bulanım mantık temelli sistem geliştirmiş ve böylece konuşlandırma ve o pozisyonda tutma sırasında harcanan yakıt üç misli, yaklaşma sırasında tüketilen yakıt bir buçuk misli azaltılmıştır.

3.4.1. Bulanık kümeler ve üyelik fonksiyonları

Bulanık mantık, sayıların komşuluğu felsefesine dayanır. Karar sürecinde bir durum bir sayıyla ifade ediliyorsa, söz konusu durumun kabul edilirliği o sayının gerçekleşmesinde sağlanacaktır. Ancak söz konusu sayıya yakın sayılar karar sürecinin bir parçası olarak algılanmayacaktır. Oysa belirli bir güven katsayısında bu sayıların farklı popülasyonların üyeleri olduğunu öne sürmek de istatistiksel açıdan yanlış olmayacaktır. Örneğin bir tezgahta işlenen bir parçanın sıcaklığının 39 C'ye ulaşması, tezgahın bakım sürecini başlatan bir durumsa belki de sıcaklığın 36 C'ye ulaşması da aynı bakım sürecinin başlaması için bir ön şart olarak kabul edilebilir. Bu durumda aynı temel amaca hizmet eden sayıların komşuluğundan söz etmek mümkündür.

X evrensel tanım kümesi üzerinde A bulanık kümesi, *X* uzayından birim aralığa bir dönüşüm olan $\mu_A(x)$ üyelik fonksiyonları ile tanımlanır: Φ(*X*) ile X uzayındaki tüm bulanık kümeler gösterilir.

3.4.2. Sonlu ve sonsuz bulanık kümeler

Bulanık kümeler sonlu ve sonsuz olabilirler. Sonlu bir X ={x_1,.....,x_n} kümesi için F sonlu bulanık kümesi

$$F = \mu_F(x_1)/x_1 + \ldots + \mu_F(x_n)/x_n = \sum_{i=1}^{n} \mu_F(x_i)/x_i$$

İfadesi ile belirlenmektedir. X sonsuz olduğunda ise;

$$F = \int_{i=1}^{X} (\mu_F(x)/x).\, d_x$$

şeklinde belirlenir.

∑: Toplam simgesi, Bulanık tekliklerin kesikli evrende bir araya getirilmesini,

∫ : İntegral simgesi, bulanık tekliklerin sürekli evrende bir araya getirilmesini,

/ **:** Bu simge, matematiksel olarak (x,μ $A(x)$) tekliğini ifade etmek için kullanılan bir ayıraçtır.

+ **:** İşareti ise, bulanık tekliklerin birleşimini gösterir.

Bulanık küme kuramı, bir elemanın bir kümeye kısmi üyeliğine izin verir ve küme üyeliği için [0,1] arasındaki herhangi bir değeri kabul eder. Eğer üyelik derecesi olarak adlandırılan üyelik fonksiyonunun değeri, 1'e eşitse x elemanının bulanık kümenin tam üyesi olduğu; eğer bu değer sıfır ise, x elemanının bulanık kümeye ait olmadığı; eğer üyelik derecesi sıfır ile 1 arasında ise x elemanının bulanık kümeye kısmen üye olduğu anlaşılır. Bu durumda üyelik fonksiyonunu aşağıdaki gibi tanımlayabiliriz [40].

Eğer A $\mathbb{R}$' de tanımlı bir kümenin elemanı ise $\mu_A(x)$ üyelik fonksiyonu $\mathbb{R} \rightarrow [0,1]$ aralığında oluşur diğer bir deyişle A kümesi A=[a_1 a_3] aralığında ise genel olarak $\mu_A(x)$ üyelik fonksiyonu aşağıdaki şekillerdeki formül ile gösterilebilir.

$$\mu_A(x) = \begin{cases} 0, & x < a_1 \\ 1, & a_1 \leq x \leq a_3 \\ 0, & x > a_3 \end{cases}$$

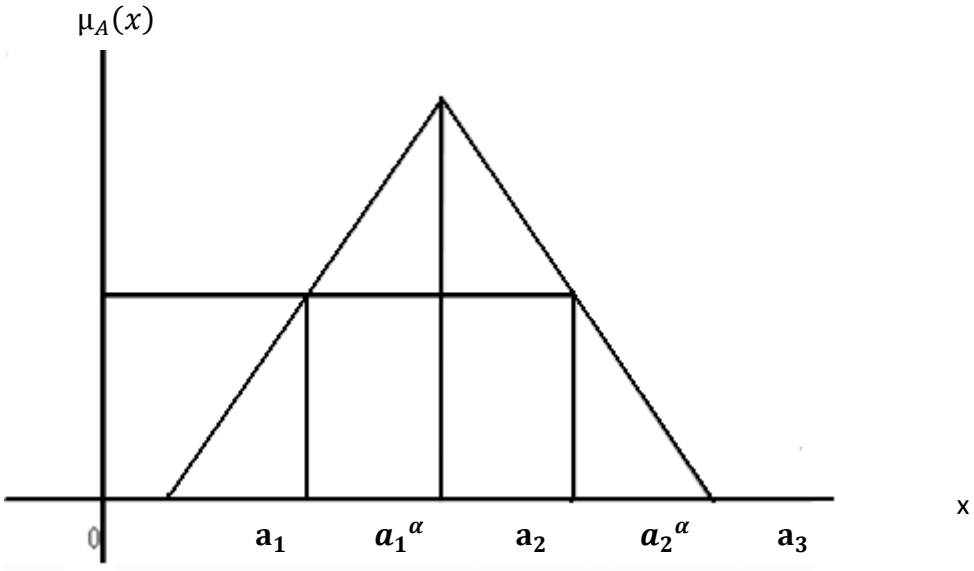
Üyelik fonksiyonları genellikle, üçgensel üyelik fonksiyonları ve yamuk üyelik fonksiyonları olmak üzere iki başlık altında incelenmektedir.

Üçgensel üyelik fonksiyonu

$$\mu_A(x) = \begin{cases} 0, & x < a_1 \\ \dfrac{x - a_1}{a_2 - a_1}, & a_1 \leq x \leq a_2 \\ \dfrac{a_3 - x}{a_3 - a_2}, & a_2 < x \leq a_3 \\ 0, & x > a_3 \end{cases}$$

formülüne göre küme A= (a_1, $a_{2,}$ a_3) a olmalıdır. Burada a_2 normal değerli üyelik olarak tanımlanabilir. Bulanık Mantık bu noktada bir α katsayısına bağlı olarak $a_{2,}$ ye yakın değerlerin, bu değere yüklenen anlam ile temsil edileceğini varsaymaktadır. Diğer bir deyişle a_2 deki belirsizlik, varsayılacak ya da dağılıma göre bulunabilecek bir α katsayısı ile tolere edilebilir [42].

Söz konusu komşuluk aşağıdaki şekil 3.30. da gösterilmiştir.



Şekil 3.31. Üçgensel Üyelik Fonksiyonu

α değeri bulanık mantık terminolojisinde kesim katsayısı olarak adlandırılır. ${a_1}^{\alpha}$ ve ${a_3}^{\alpha}$ sayıları ise a_2 normal değerinin komşuluğunu oluşturan alt ve üst sınır değerleridir [41].

α aşağıdaki formüllerle hesaplanır.

$$\alpha = \frac{{a_1}^{\alpha} - a_1}{a_2 - a_1}$$

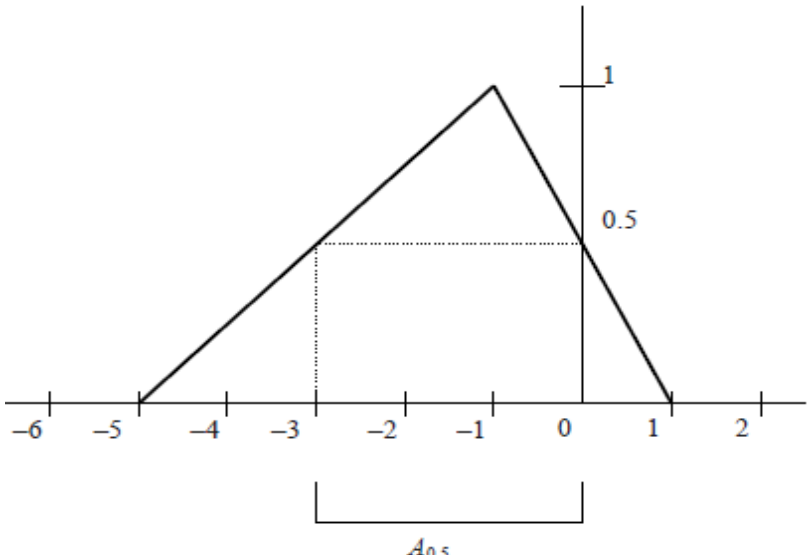
$$\alpha = \frac{a_3 - {a_3}^{\alpha}}{a_3 - a_2}$$

Örneğin üçgensel bulanık mantık sayılarına ilişkin küme A = (–5,–1,1) ise bu durumda yukarıdaki formülden üyelik fonksiyonu hesaplanır.

$$\mu_A(x) = \begin{cases} 0, & x < -5 \\ \dfrac{x+5}{4}, & -5 \le x \le -1 \\ \dfrac{1-x}{2}, & -1 < x \le 1 \\ 0, & x > 1 \end{cases}$$

Eğer karar verici α kesim katsayısını 0,5 olarak saptamışsa -1 normal değerinin komşuları, formüller vasıtasıyla $a_1{}^{0,5} = -3$ ve $a_3{}^{0,5} = 0$ olarak bulunacaktır.

Diğer bir deyişle -1 normal değeri ile aynı anlam düzeyinde bulunan sayılar kümesi $[-3,0]$ aralığıdır. Söz konusu şekil 3.31. de gösterilmiştir.



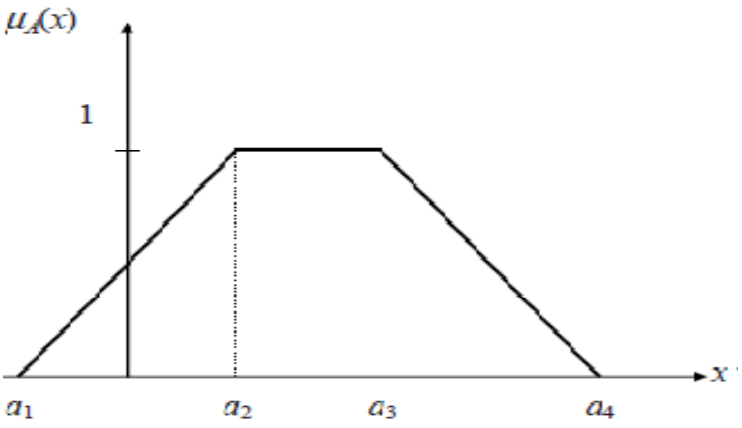
Şekil 3.32. Üçgensel Üyelik Fonksiyonu Örnek Şekli

Yamuk üyelik fonksiyonu

Eğer bulanık mantık sayılarına ilişkin kümede normal kabul edilen iki değer varsa, yani A= (a_1, a_2, a_3, a_4) şeklinde 4 belirleyici değerden oluşuyorsa bu durumda üyelik fonksiyonu yamuk üyelik fonksiyonu tipinde oluşacaktır. Yamuk üyelik fonksiyonu aşağıdaki gösterilmiştir.

$$\mu_A(x) = \begin{cases} 0, & x < a_1 \\ \dfrac{x - a_1}{a_2 - a_1}, & a_1 \le x \le a_2 \\ 1, & a_2 < x \le a_3 \\ \dfrac{a_4 - x}{a_4 - a_3}, & a_3 < x \le a_4 \\ 0, & x > a_4 \end{cases}$$

Söz konusu fonksiyona ait şekil aşağıdaki gibidir.



Şekil 3.33. Yamuk Üyelik Fonksiyonu

Yukarıda gösterilmiş olan fonksiyonlar farklı tarzda çalışılmış implikasyonlara sahiptir. Aşağıda bu implikasyonlar gösterilecektir.

Dienes – Rescher İmplikasyonu

$$\mu_{Q_D}(x,y) = \max\left[1 - \mu_{FP_1}(x), \mu_{FP_2}(y)\right]$$

Lukasiewicz implikasyonu

$$\mu_{Q_L}(x,y) = \min\left[1, 1 - \mu_{FP_1}(x) + \mu_{FP_2}(y)\right]$$

Zadeh implikasyonu

$$\mu_{Q_Z}(x,y) = \max\left(\min\left[\mu_{FP_1}(x), \mu_{FP_2}(y)\right], 1 - \mu_{FP_1}(x)\right)$$

Gödel implikasyonu

$$\mu_{\mathrm{Q_G}}(\mathrm{x},\mathrm{y}) = \begin{cases} 1, & \mu_{\mathrm{FP_1}}(\mathrm{x}) \le \mu_{FP_2}(y) \\ \mu_{\mathrm{FP_2}}(\mathrm{y}), & diğer\ durumlarda \end{cases}$$

Mamdani implikasyonu

$$\mu_{Q_M}(x,y) = \left(\min\left[\mu_{FP_1}(x).\,\mu_{FP_2}(y)\right]\right.$$

Goguen implikasyonu

$$\mu_{\mathrm{Q_{Gog}}}(\mathrm{x},\mathrm{y}) = \begin{cases} 1, & \mu_{\mathrm{FP_1}}(\mathrm{x}) = 0 \\ \min(\,1, \mu_{\mathrm{FP_2}}(\mathrm{y})/\mu_{\mathrm{FP_1}}(\mathrm{x}), & diğer\ durumlarda \end{cases}$$

Reichenbach implikasyonu

$$\mu_{Q_{Rei}}(x,y) = (1 - \mu_{FP_1}(x) + \mu_{FP_1}(x).\,\mu_{FP_2}(y)$$

Larsen implikasyonu

$$\mu_{Q_{Lar}}(x,y) = \mu_{FP_1}(x).\,\mu_{FP_2}(y)$$

Örnek: x_1 arabanın hızı x_2 ivme ve y ivmeye uygulanan güç olsun. Bu durumda "Yavaş" bulanık kümesi belirlenmiştir ve matematiksel olarak aşağıdaki gibi yazılabilir.

$$\mu(x_1) = \begin{cases} 1, & x_1 < 50 \\ \frac{80 - x_1}{30}, & 50 \leq x_1 \leq 80 \\ 0, & x_1 > 80 \end{cases}$$

Küçük" ivme alanında bir bulanık kümedir ve onun üyelik fonksiyonu ise aşağıdaki gibidir

$$\mu(x_2) = \begin{cases} \frac{10 - x_2}{10}, & 0 \leq x_2 \leq 10 \\ 0, & x_2 > 10 \end{cases}$$

Uygulanan güç te bir bulanık kümedir ve onun üyelik fonksiyonu aşağıdaki gibidir.

$$\mu(y) = \begin{cases} 0, & y < 1 \\ y - 1, & 1 \leq y \leq 2 \\ 1, & y > 2 \end{cases}$$

x_1, x_2 ve y'nin alanları uygun olarak $U_1 = [0, 160]$, $U_2 = [0, 30]$ ve $V = [0,3]$ olsun

$\mu_{FP_1}(x) = \mu(x_1).\mu(x_2)$

$$\mu_{FP_1}(x) = \begin{cases} 0, & x_1 > 80 \; veya \; x_2 > 10 \\ \frac{10 - x_2}{10}, & x_1 \leq 50 \text{ veya } x_2 \leq 10 \\ \left(\frac{80 - x_1}{30}\right).\left(\frac{10 - x_2}{10}\right), & 50 < x_1 \leq 80 \text{ veya } x_2 < 10 \end{cases}$$

Dienes – Rescher İmplikasyonu uygulandığında

$$\mu_{Q_D}(x, y) = \max\left[1 - \mu_{FP_1}(x), \mu_{FP_2}(y)\right]$$

$$1-\mu_{FP_1}(x)=\begin{cases}1, & x_1>80\ veya\ x_2>10\\ \frac{x_2}{10}, & x_1\le 50 \text{ veya } x_2\le 10\\ 1-\frac{(80-x_1)(10-x_2)}{300}, & 50<x_1\le 80 \text{ veya } x_2<10\end{cases}$$

$$\mu_{FP_2}(y)=\begin{cases}0, & y<1\\ y-1, & 1\le y\le 2\\ 1, & y>2\end{cases}$$

$$\mu_{Q_D}(x,y)=\begin{cases}1, & x_1>80\ veya\ x_2>10 \text{ veya } y>2\\ \frac{x_2}{10}, & x_1\le 50 \text{ ve } x_2\le 10 \text{ ve } y<1\\ 1-\frac{(80-x_1)(10-x_2)}{300}, & 50<x_1\le 80 \text{ ve } x_2<10 \text{ ve } y<1\\ (y-1, x2/10), & x_1<50\ ve\ 0\le x_2\le 10\ ve\ 1\le y\le 2\\ \left(y-1{,}1-\frac{(80-x_1)(10-x_2)}{300}\right), & 50\le x_1\le 80 \text{ ve } x_2<10 \text{ ve} 1\le y\le 2\end{cases}$$

Şeklinde fonksiyon oluşur.

3.4.3. Bulanık sistem

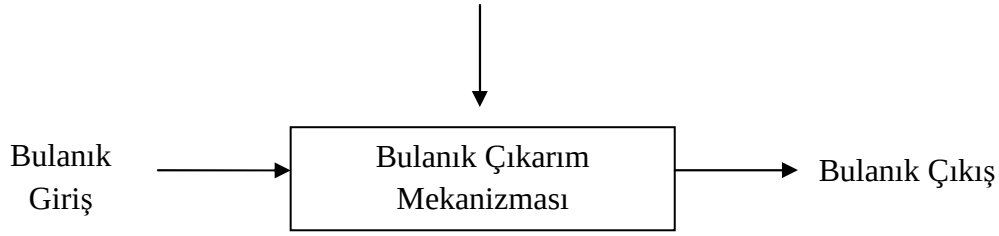
Bulanık kümeler ve bulanık mantık teorisinin en etkin uygulama alanı kontrol sistemleridir. Geleneksel kontrol sistemleri bulanık teorinin yardımıyla bulanık kontrol sistemlerine dönüştürülebilir ve böyle sistemlerin uygulanması birçok avantajı beraberinde getirir.

Genelde, bulanık sistemler bilgiye dayalı veya kurala dayalı sistemlerdir. Yani bir bulanık sistemin temelinde “Eğer - O halde” kuralları durmaktadır. Örneğin, aşağıdaki kuralı bulanık sistemin bir kuralı hesap etmek yerinde olur. Eğer sıcaklık soğuk ve basınç düşük ise, o halde sıcak su supabını orta pozitifte tut ve soğuk su supabının durumunu değiştirme [43].

Burada soğuk, düşük, orta pozitif gibi dilsel değerler kullanılır ve bu dilsel değerlerin uygun üyelik dereceleri mevcuttur. Bir bulanık sistem tasarlanmasına karar

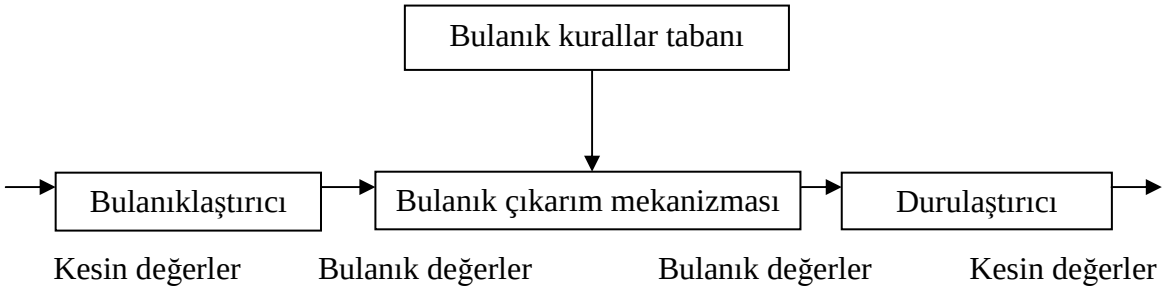
verildikten sonra ilk yapılacak iş Eğer – o halde kurallar toplusunu elde etmektir. Bu kurallar çoğu zaman uzmandan yararlanılarak toplanılır.

Literatürde genelde üç tip bulanık sistemden söz edilmektedir. 1) temiz (pure) bulanık sistemler; 2) bulanıklaştırıcılı ve durulaştırıcılı sistemler; 3) Takagi – Sugeno – Kang (TSK) bulanık sistemler. Temiz bulanık sistemlerde sistemin giriş ve çıkışları bulanıktır. Bulanık çıkarım mekanizması (Fuzzy Inference Engine) bulanık girişlere uygun kuralları bulanık kurallar tabanından (Fuzzy Rule Base) alarak imal edilir ve vardığı sonuçta bulanık olur [43].



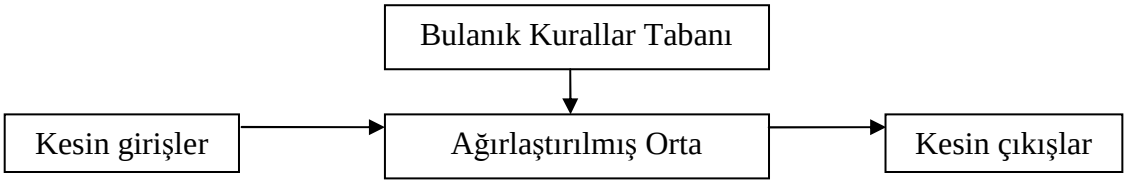
Şekil. 3.34. Temiz Bulanık Sistemler

Temiz bulanık sistemlerde giriş ve çıkış değerleri dilsel olarak kullanılmaktadır, halbuki gerçek sistemlerde bu değerler kesindir, bulanık değildir. Bu dezavantajı kaldırmak için sisteme girişteki kesin değerleri bulanık değerlere dönüştüren bulanıklaştırıcı (fuzzitier) ve çıkıştaki bulanık değerleri kesin değerlere dönüştüren durulaştırıcı yı (deffuzzitier) uygun olarak sistemin girişine ve çıkışına ilave edilirler. Böyle sistemler bulanıklaştırıcılı ve durulaştırıcılı sistem olarak adlandırılmaktadır. Aşağıda şekil 3.34. de gösterilmiştir.



Şekil 3.35. Bulanıklaştırıcılı ve Durulaştırıcılı Sistem

Bulanık sistemlerin TSK modelinde sistemin giriş ve çıkış kesin değerlerdir. Bu modelde bulanık çıkarım mekanizması yerine ağırlaştırılmış orta (weighted average) kullanılır (Şekil 4.3). Burada ağırlaştırılmış orta; Eğer – o halde kuralının o halde kısmında çoğu zaman bir matematik formüllerinin temsilidir.



Şekil 3.36. Takagi – Sugeno – Kang (TSK) bulanık sistemler

Böyle bir kullanım ise bulanık mantığın çeşitli prensiplerini uygulamaya imkan vermemektedir. Bu yüzden TSK modelinin uygulama alanı kısıtlıdır.

Bulanık teori, bulanık sistemlerde, özellikle de otomatik kontrol sistemlerinde insan bilgisine dayanan dilsel bir kontrol strateji uygulamak için kullanılır. bulanık kontrol sistemleri tasarlarken sırasıyla hedef, bilgi tabanını oluşturan bulanık kontrol kuralları belirlenir ve bulanıklaştırma ve durulaştırma yapılır. Bulanık teorinin ileri sürülmesinden (1965) kısa bir süre geçmesine rağmen bulanık kontrol çok çabuk bir gelişme sağlamıştır. Bu gelişmenin önemli evreleri aşağıdaki tabloda gösterilmektedir [Bkz: Lee, C.L., Fuzzy Logic In Control Systems: Fuzzy Logic Vol.20, controller, Part1 and Part2., IEEE Trans. On systems and Cybernetics.]

Tablo 3.6. Bulanık Teori Gelişim

Yıl	Uygulayan	Uygulama alanı
1972	Zadeh	Bulanık kontrolün öne sürülmesi
1973	Zadeh	Dilsel yaklaşım
1974	Mamdani ve Assilian	Buhar motoru kontrolü
1976	Rutherford	Kontrol algoritmalarının analizi
1977	Ostergaard	Isı değiştirici ve çimento fırın kontrolü
1977	Willaeys	Optimal bulanık kontrol
1979	Komolov	Sonlu otomasyon
1980	Tong	Atık suyun değerlendirilmesi
1980	Fukami, Mizumoto ve Tanaka	Bulanık koşul çıkarımı
1983	Hirota ve Pedrycz	Olasılıklı bulanık kümeler
1983	Takagi ve Sugeno	Bulanık kontrolün türetilmesi
1983	Yasunubo ve Miyamato	Tahmini bulanık kontrol
1984	Sageno ve Murakabi	Bir arabanın park etme kontrolü
1985	Kiszka ve Gupta	Bulanık sistemin kararlılığı
1985	Togai ve Watanabe	Bulanık yonga
1986	Yamakawa	Bulanık kontrolör donanım sistemi
1987	Dubois ve Parade	Sendai metrosunda uygulama
1988		Yaklaşık muhakeme
1990- …		Optimum sistem planlaması
		Yapay zeka
		Uzman sistemler
		Kontrol teorisi
		Kalite kontrol,
		Çok amaçlı karar verme
		Ürün planlaması, seçimi
		Taşıma, ulaşım
		Oyunlar kuramı
		Çevre yönetimi
2009		Bankacılık finansı

3.4.4. Bulanık küme teorisi ve bulanık doğrusal optimizasyon

Bir olay veya bir sistem miktarı karakteristiklerle sunulmayınca bu olayın veya sistemin iyi anlaşıldığı sayılamaz olması modern bilimin temel prensiplerinden biridir. Bu açıdan bakıldığında bilimsel bilginin özünü teşkil eden bileşenlerin çoğuna, bu bileşenlerin davranışları hakkında miktarı enformasyon almaya olanak tanıyan ve çeşitli sistemlerin matematik modellerini oluşturmak için gereken prensip ve yöntemler toplusu gibi bakılabilir.

Bilgisayarların geniş kullanım alanına sahip olması ile insan bilgisinin birçok alanında, yöntemler daha hızlı yaygınlaşmaya başladı. Burada bilgisayarların hiç şüphesiz, davranışları mekanik, fizik, kimya ve elektromagnetizma kanunları ile belirlenen mekanistik sistemlere uygulanışı çok efektif olmuştur ve olmaktadır. Maalesef, aynı şeyi humanistik (insancıl) sistemler hakkında söyleyememekteyiz. (Humanistik sistemler, davranışına insanın muhakemelerinin, çevreyi algılamasının veya emosyonlarının etki yaptığı sistemlerdir. Örneğin; ekonomik, politik, hukuk, eğitim vb. sistemleri) [44].

Bu sistemler şimdiye kadar matematik analize ve bilgisayar modellemeye büyük direnç göstermişlerdir. Humanistik sistemlere uygulamadaki başarısızlık, hesaplamalardaki yüksek hassasiyet isteğinin ve bunun yüksek hızla yapılması isteğinin uyuşmazlığından ileri geldiği bazı bilim adamları tarafından gösterilmektedir. Diğer bir deyimle, sistemin karmaşıklığı ve bu karmaşıklığı analiz etmek için kullanılan hassasiyet ters orantılıdır. Buradan, humanistik sistemlerin davranışı hakkında önemli sonuçlar alabilmek için hesaplamalardaki yüksek hassasiyet ve kesinlikten kaçınmak gerektiği sonucuna varılabilir.

Gerçek dünya karmaşıktır. Bu karmaşıklık genel olarak belirsizlik, kesin düşünceden yoksunluk ve karar verilemeyişten kaynaklanır. Birçok sosyal, ekonomik ve teknik konularda insan düşüncesinin tam anlamıyla olgunlaşmamış oluşundan dolayı belirsizlikler her zaman bulunur. İnsan tarafından geliştirilmiş olan bilgisayarlar, bu türlü bilgileri işleyemezler ve çalışmaları için sayısal bilgiler gereklidir. Gerçek bir olayın kavranılması insan bilgisinin yetersizliği ile tam anlamıyla mümkün

olmadığından, insan, düşünce sisteminde ve zihninde bu gibi olayları yaklaşık olarak canlandırarak yorumlarda bulunur.

Genel olarak değişik biçimlerde ortaya çıkan karmaşıklık ve belirsizlik gibi tam ve kesin olmayan bilgi kaynaklarına bulanık kaynaklar adı verilir. Zadeh tarafından gerçek dünya sorunları ne kadar yakından incelemeye alınırsa, çözümün daha da bulanık hale geleceği ifade edilmiştir. Çünkü çok fazla olan bilgi kaynaklarının tümünü insan aynı anda ve etkileşimli olarak kavrayamaz ve bunlardan kesin sonuçlar çıkaramaz.

3.4.5. Bulanık doğrusal optimizasyon yaklaşımları

Doğrusal optimizasyon modellerindeki bulanıklık, amaç fonksiyonu ve kısıtlayıcı katsayılarının tam olarak bilinmediği ve modeldeki bazı eşitsizlikler (veya eşitlikler) için net olmayan sınırların tanımlanabileceği anlamına gelir. Bunlar, eksik bilgiden veya yapısal durumdan kaynaklanabilir. Örneğin; "maliyeti olabildiğince azaltmak" şeklindeki bir hedef, bulanık bir amacı niteler. Diğer bir değişle, amaç fonksiyonu için belirlenen erişim düzeyi bulanık olarak ifade edilir. Benzer olarak, $Ax \leq b$ şeklinde ifade edilen kısıtlayıcı kümesinde, $\leq$ işaretinin matematiksel anlamına belirli bir aralıkta tolerans gösterilebilir. Bir ürünün satış fiyatının, dolayısıyla elde edilebilecek birim karın, rekabet, maliyet vb. faktörler nedeniyle kesin olarak ifade edilmesi gerçekçi olmayabilir. Diğer yandan belli bir ürüne olan talep miktarı çoğu durumda tam olarak bilinemez.

Literatürde bulanık doğrusal programlama modeliyle ilgili birçok yaklaşım vardır. Ancak bu modellerin çıkış noktası Zimmermann çözüm modelidir.

Zimmermann yaklaşımı

Zimmermann, bulanık amaç ve bulanık kısıtlayıcılı doğrusal programlama problemleri için simetrik bir yaklaşım önermiştir. Zimmermann'a göre, bulanık amaç fonksiyonu karar vericiden sağlanan bulanık bir erişim düzeyi ile bulanık bir kısıtlayıcı olarak ifade edilebilir. Bu durumda, bulanık karar kümesi belirlenirken

bulanık amaç ve bulanık kısıtlayıcılar birbirinden farksız olarak ele alınır. nolu denklem ile gösterilen bulanık amaç ve bulanık kısıtlayıcılı bir doğrusal optimizasyon problemi, aşağıda verilen kısıtlayıcı kümesinden çözüm vektörü x'in bulanması şeklinde ifade edilir [46].

$$\text{Min } Z = c^T x \underset{\sim}{\leq} b_0$$

$$(A_x)_i \underset{\sim}{<} b_i \quad i = 1, 2, \dots, m$$

$$x_i \geq 0$$

Burada kısıtlardaki $\underset{\sim}{\leq}$ işareti, $\leq$ işaretinin bulanıklaştırılmış halidir. $(A_x)_i$ kısıtlayıcısı b_i civarında veya daha azdır şeklinde okunur. Benzer olarak amaç fonksiyonunda da aynı işaret vardır, yine burada da $c^T x$ amacı b_0 civarında veya daha azdır şeklinde yorumlanır. Amaç fonksiyonunun her iki tarafı (-1) ile çarpılırsa, bulanık doğrusal programlama problemi aşağıdaki gibi ifade edilebilir.

$$\text{Min } Z = c^T x \underset{\sim}{\geq} b_0$$

$$(A_x)_i \underset{\sim}{\leq} b_i \quad i = 1, 2, \dots, m$$

$$x_i \geq 0$$

Burada $B = \begin{bmatrix} -c^T \\ A_i \end{bmatrix}$ ve $e = \begin{bmatrix} -b_0 \\ b_i \end{bmatrix}$ sütun vektörleri şeklinde tanımlanarak aşağıda ki gibi düzenlenebilir.

$$B_x \geq e$$

$$x \geq 0$$

Bulanık amaç ve bulanık kısıtlayıcılar, bulanık kümeler olarak tanımlandığı için, bunlara ilişkin üyelik fonksiyonlarının belirlenmesi gerekir. Yukarıdaki modelin i'ninci satırı için, üyelik fonksiyonunun tek düze olarak artmayan bir yapıda olması gerekir. Yani, i'ninci bulanık eşitsizlik tamamen karşılanıyorsa, üyelik derecesi 1 olmalı; i'ninci bulanık eşitsizlik tamamen karşılamıyorsa, üyelik derecesi 0 olmalıdır

$e_i = b_i$ i=(0, 1, 2, …, m) terimi i'ninci bulanık eşitsizlik için karar vericinin ulaşmak istediği erişim düzeyi; p_i terimi ise i'ninci erişim düzeyi için karar vericinin belirlediği maksimum tolerans miktarı olmak üzere $[b_i, b_i + p_i]$ aralığında üyelik derecesi 1'den 0 a doğru tek düze olarak azalmalıdır.

p_i 'ler amaç fonksiyonu ve kısıtlayıcılardaki kabul edilebilir toleransları gösteren ve karar verici tarafından belirlenen sabitlerdir. Bu durumda, i'ninci bulanık eşitsizliğin üyelik fonksiyonu aşağıdaki gibi ifade edilebilir.

$$\mu_i[(\beta x)_i] = \begin{cases} 0, & (\beta x)_i \geq b_i + p_i \\ 0 - 1, & b_i < (\beta x)_i \leq b_i + p_i \\ 1, & (\beta x)_i \leq b_i \end{cases}$$

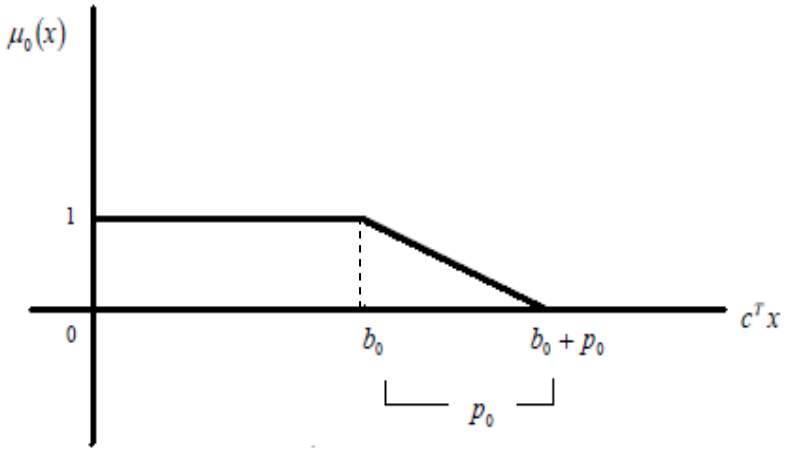
Bulanık kısıtlayıcıların parçalı doğrusal üyelik fonksiyonları

$$\mu_0(x) = \begin{cases} 0, & c^T x \geq b_0 + p_0 \\ 1 - \dfrac{c^T x - b_0}{p_0}, & b_0 < c^T x \leq b_0 + p_0 \\ 1, & c^T x \leq b_0 \end{cases}$$

$$\mu_i(x) = \begin{cases} 0, & (A_x)_i \geq b_i + p_i \\ 1 - \dfrac{(A_x)_i - b_i}{p_i}, & b_i < (A_x)_i \leq b_i + p_i \\ 1, & (A_x)_i \leq b_i \end{cases}$$

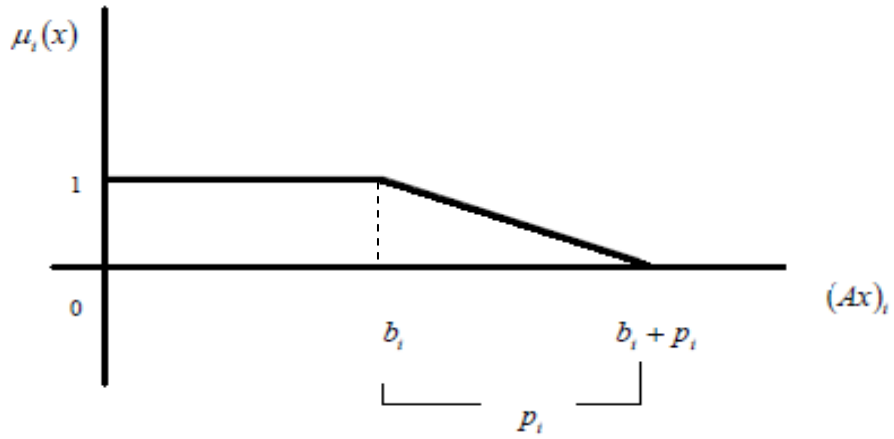
Bulanık amaç ve bulanık kısıtlayıcıların üyelik fonksiyonları aşağıdaki şekillerde gösterilmiştir. Bu şekillerde, bulanık amaç fonksiyonu ve bulanık kısıtlayıcılara

ilişkin üyelik fonksiyonlarının sırasıyla tek düze olarak artmayan fonksiyonlar olduğu görülebilmektedir.



Şekil 3.37. $c^T x \leq b_0$ Eşitsizliğin Üyelik Fonksiyonu

~



Şekil 3.38. $(A_x)_i \leq b_i$ Eşitsizliğin Uyelik Fonksiyonu

~

Bulanık amaç ve bulanık kısıtlayıcıların üyelik fonksiyonları belirlendiği için, bulanık karar kümesi aşağıda verilen ifadeden oluşturulabilir.

$$\mu_D(x) = \min [\mu_0(x), \mu_i(x)] \quad i = 1, 2, \dots, m$$

~

Bulanık karar kümesinin en yüksek üyelik dereceli elemanı;

$$\mu_D(x^M) = \max (\min[\mu_0(x), \mu_i(x)]) \quad i = 1, 2, \dots, m \text{ olur.}$$

~

Yukarıda gösterilen formülüzasyon detaylandırıldığında

$\mu_D(x^M) = \max\left(\min\left[\left(1 - \frac{c^T x - b_0}{p_0}\right).\left(1 - \frac{(A_x)_i - b_i}{p_i}\right)\right]\right)$ i = 1, 2, … , m elde edilir.

~

$\mu_D(x^M)$; Bulanık amacı, bulanık kısıtlayıcılar için tolerans belirtildiği zaman yani doğrusal bir model kurulduğu zaman belirlenebilir. Bunun için, bulanık karar kümesinin λ değişkeni ile gösterilmesi gerekir. Burada λ değişkeni, bulanık amaç ve bulanık kısıtlayıcıların çözüm vektörü x tarafından karşılanma derecesini gösterir. λ değişkeni [0,1] aralığında tanımlanır. Bu noktadan hareketle

$\mu_0(x) \geq \lambda$

$\mu_i(x) \geq \lambda$

Maks λ

Kısıtlar

$\mu_0(x) \geq \lambda$

$\mu_i(x) \geq \lambda$

λ ϵ [0,1]

Bulanık amaç ve bulanık kısıtlayıcıların üyelik fonksiyonları, modelde yerine konduğu zaman, aşağıdaki doğrusal optimizasyon modeline ulaşılır.

Maks λ

$$1 - \frac{c^T x - b_0}{p_0} \geq \lambda$$

$$1 - \frac{(A_x)_i - b_i}{p_i} \geq \lambda$$

λ ϵ [0,1]

$x \geq 0$

Maks λ

$c^T x \geq b_0 + (1-\lambda).p_0$

$(A_x)_i \leq b_i + (1-\lambda).p_i$

$\lambda \epsilon [0,1]$

$x \geq 0$

Chanas yaklaşımı

Chanas, bulanık kısıtlayıcıların belirlediği uygun çözüm alanının bilgi eksikliği yüzünden, amaç fonksiyonuna ilişkin erişim düzeyi ile tolerans miktarının karar verici tarafından başlangıçta belirlenemeyeceğini öne sürmüştür. Chanas, bulanık amaç fonksiyonlu ve bulanık kısıtlayıcılı doğrusal programlama problemleri için, parametrik programlama sonrası karar verici tercihini dikkate alan bir çözüm yaklaşımı önermiştir [47].

$Z_{maks} = c^T \tilde{x}$

$(A_x)_i \leq b_i \quad i = 1, 2, \ldots, m$

$x \geq 0$

Bu modelde, amaç fonksiyonundaki katsayılar bulanık sayılarla veya bulanıklığı belirten tolerans aralıkları ile tanımlanır.

Werners yaklaşımı

Bir diğer yaklaşım Werners yaklaşımıdır. Werners'e göre, bulanık kısıtlayıcılı doğrusal optimizasyon problemlerinde kısıtlayıcıların bulanık olması, amaç fonksiyonunda bulanık olmasını gerektirir. Bu yaklaşım aşağıdaki gibi modellenir [48].

$$Z_{maks} = c^T x$$

$$(A_x)_i \leq b_i \qquad i = 1, 2, \dots, m$$

$$x \geq 0$$

Verdegay yaklaşımı

Verdegay, bulanık amaç katsayılı doğrusal optimizasyon problemlerinin dualiteden dolayı, bulanık kısıtlayıcılı doğrusal optimizasyon problemi olarak çözülebileceğini öne sürmüştür [49]. Yaklaşım şu şekildedir.

$$Z_{maks} = \sum_{j=1}^{n} c_j x_j$$

$$\sum_{j=1}^{n} a_{ij}.x_j \leq b_i \qquad i = 1, 2, \dots, m$$

$$x_j \geq 0$$

Wang ve Liang yaklaşımı

Wang ve Liang da 2004 yılında yayınladıkları makalelerinde amaç, kısıtlar ve tüm katsayıların bulanıklaştırılarak doğrusal optimizasyon probleminin çözüleceği yaklaşımını savunmuşlardır [50].

$$\underset{\sim}{Z}\text{maks} = \sum_{j=1}^{n} c_j \underset{\sim}{x_j}$$

$$\sum_{j=1}^{n} a_{ij} . \underset{\sim}{x_j} \leq \underset{\sim}{b_i} \qquad i = 1, 2, \dots, m$$

$$x_j \geq 0$$

Yukarda gösterilen yaklaşımlar aşağıda özetlenen durumların varyasyonları şeklinde oluşmuştur.

Bulanık doğrusal optimizasyon modellerinin tasarımında göz önüne alınan 3 temel bulanıklılık vardır [51].

Sağ taraf sabitleri bulanık olabilir.

$$Z = c^T x$$

$$(A_x)_i \leq \underset{\sim}{b_i} \qquad i = 1, 2, \dots, m$$

$$x \geq 0$$

Amaç fonksiyonunun katsayısı bulanık olabilir.

$$Z = c^T \tilde{x}$$

$$(A_x)_i \leq b_i \qquad i = 1, 2, \dots, m$$

$$x \geq 0$$

Teknolojik katsayı bulanık olabilir.

$$Z = c^T x$$

$$(\tilde{A}_x)_i \leq b_i \qquad i = 1, 2, \dots, m$$

$$x \geq 0$$

3.4.6. Bir problem üzerinden bulanık optimizasyon ve doğrusal optimizasyon tekniğinin karşılaştırılması

Örnek:

Wang ve Liang 2004' te yayınladıkları çalışmalarında bütünleşik üretim planlama problemleri için çok ürünlü ve 3 ile 18 dönemi kapsayacak bir planlama ufkunu öngören envanter seviyesi, işgücü seviyesi, fazla mesai, makine, stok alanı kısıtlarını dikkate alarak toplam üretim maliyetini minimize etmeyi hedefleyen detaylı bir model geliştirmiştir. Verilen örnekte Wang ve Liang (2004)' ın geliştirdiği modelden yararlanılmıştır [50].

Belirli bir müşteriye ait 10 ürün üzerinden bir yıl içindeki belirli 4 aya ait talep, işgücü, makine kapasite, üretim miktarı, envanter seviyesi gibi kontrol edilebilir değişkenlerle aşağıda modeli oluşturularak karar değişkenleri, kısıtlayıcılar ve amaç fonksiyonunu belirlenmiştir.

Çözüm

N : ürün tipleri

T : Planlama süreci

D_{nt} : n. ürüne ait t. zamandaki tahmini talep (adet)

L_{nt} : n. ürüne ait t. zamandaki birim normal mesai üretim maliyeti ($ / adet)

Q_{nt} : n. ürüne ait t. zamandaki normal mesai üretim adedi (adet)

o_{nt} : n. ürüne ait t. zamandaki birim fazla mesai maliyeti ($ / adet)

co_{nt} : n. ürüne ait t. zamandaki fazla mesai üretim miktarı (adet)

h_{nt} : n. ürüne ait t. zamandaki birim envanter taşıma maliyeti ($ / adet)

S_{nt} : n. ürüne ait t. zamandaki stok seviyesi (adet)

k_t : t. zamandaki işe alma maliyeti ($ / adam.dak)

H_t : t. zamandaki işe alınan işçi miktarı (adam.dak)

u_t : t. zamandaki işten çıkarma maliyeti ($ / adam.dak)

uk_t : t. zamandaki işten çıkarılan işçi miktarı (adam.dak)

- **KISITLAR**

Stok Taşıma Kısıtı

$$S_{nt-1} + Q_{nt} + co_{nt} - S_{nt} = D_{nt}$$

Gerçek dünyada bütünleşik üretim planlama problemlerinde tahmini talep, piyasanın dinamik olması nedeniyle değişkenlik göstermekte, kesin olarak bilinememektedir. Normal ve fazla mesai üretim miktarı ile stok seviyelerinin toplamı market talebine eşit seviyede olmalıdır. Yıl içindeki 4 aylık dönemlere ait 10 ürün için piyasa talebi, talep toleransları ile alt ve üst talep sınır değerleri aşağıda gösterilen Tablo 3.7. Tablo 3.8. Tablo3.9. ve Tablo3.10. da ki gibidir.

Tablo 3.7. Piyasa Talebi Değerleri

Adet/Dönem	1.Dönem	2.Dönem	3.Dönem	4.Dönem
Ürün 1	3000	3200	1600	2400
Ürün 2	2300	3000	1200	3300
Ürün 3	2600	2400	2400	9900
Ürün 4	2300	2700	2100	10200
Ürün 5	3600	4800	4500	7500
Ürün 6	4420	5040	4620	7350
Ürün 7	5700	7200	4700	1700
Ürün 8	2432	2448	2304	8712
Ürün 9	1100	900	800	4100
Ürün 10	490	490	630	2170

Tablo 3.8. Dönemlere Ait Talep Toleransları

Adet/Dönem	1.Dönem	2.Dönem	3.Dönem	4.Dönem
Ürün 1	400	400	200	200
Ürün 2	300	300	300	300
Ürün 3	300	300	300	450
Ürün 4	150	150	150	600
Ürün 5	300	300	300	450
Ürün 6	250	250	250	500
Ürün 7	200	400	200	100
Ürün 8	180	180	180	360
Ürün 9	100	100	100	200
Ürün 10	70	70	140	140

Tablo 3.9. Talep Alt Sınır (talep – tolerans)

Adet/Dönem	1.Dönem	2.Dönem	3.Dönem	4.Dönem
Ürün 1	2600	2800	2600	2200
Ürün 2	2000	2700	2000	3000
Ürün 3	2300	2100	2300	9450
Ürün 4	2150	2550	2150	9600
Ürün 5	3300	4500	3300	7050
Ürün 6	4170	4790	4170	6850
Ürün 7	5500	6800	5500	1600
Ürün 8	2252	2268	2252	8352
Ürün 9	1000	800	1000	3900
Ürün 10	420	420	420	2030

Tablo 3.10. Talep Üst Sınır (talep + tolerans)

Adet/Dönem	1.Dönem	2.Dönem	3.Dönem	4.Dönem
Ürün 1	3400	3600	1800	2600
Ürün 2	2600	3300	1500	3600
Ürün 3	2900	2700	2700	10350
Ürün 4	2450	2850	2250	10800
Ürün 5	3900	5100	4800	7950
Ürün 6	4670	5290	4870	7850
Ürün 7	5900	7600	4900	1800
Ürün 8	2612	2628	2484	9072
Ürün 9	1200	1000	900	4300
Ürün 10	560	560	770	2310

yukarıdaki tablolara göre dönemler için bulanıklaştırılmış stok kısıtları aşağıdaki gibidir.

A) Bulanık optimizasyonla çözüm

1.dönem için bulanıklaştırılmış stok kısıtları

$$S_{nt-1} + Q_{nt} + co_{nt} - S_{nt} \tilde{=} D_{nt}$$

$S10 + Q11 + co11 - S11 \tilde{=} 3000$

$S20 + Q21 + co21 - S21 \tilde{=} 2300$

$S30 + Q31 + co31 - S31 \tilde{=} 2600$

$S40 + Q41 + co41 - S41 \tilde{=} 2300$

$S50 + Q51 + co51 - S51 \tilde{=} 3600$

$S60 + Q61 + co61 - S61 \tilde{=} 4420$

$S70 + Q71 + co71 - S71 \tilde{=} 5700$

$S80 + Q81 + co81 - S81 \tilde{=} 2432$

$S90 + Q91 + co91 - S91 \tilde{=} 1100$

$S100 + Q101 + co101 - S101 \tilde{=} 490$

2.dönem için bulanıklaştırılmış stok kısıtları

$S11 + Q12 + co12 - S12 \tilde{=} 3200$

$S21 + Q22 + co22 - S22 \tilde{=} 3000$

S31 + Q32 + co32 - S32 = 2400

S41 + Q42 + co42- S42 – 2700

S51 + Q52 + co52 - S52 = 4800

S61 + Q62 + co62 - S62 = 5040

S71 + Q72 + co72 - S72 = 7200

S81 + Q82 + co82 - S82 = 2448

S91 + Q92 + co92 - S92 = 900

S101 + Q102 + co102 -S102 = 490

3.dönem için bulanıklaştırılmış stok kısıtları

S12 + Q13 + co13 - S13 = 1600

S22 + Q23 + co23 - S23 = 1200

S32 + Q33 + co33 - S33 = 2400

S42 + Q43 + co43 - S43 = 2100

S52 + Q53 + co53 - S53 = 4500

S62 + Q63 + co63 - S63 = 4620

S72 + Q73 + co73 - S73 = 4700

S82 + Q83 + co83 - S83 = 2304

S92 + Q93 + co93 - S93 = 800
~

S102 + Q103 + co103 - S103 = 630
~

4.dönem için bulanıklaştırılmış stok kısıtları

S13 + Q14 + co14 - S14 = 2400
~

S23 + Q24 + co24 - S24 = 3300
~

S33 + Q34 + co34 - S34 = 9900
~

S43 + Q44 + co44 - S44 = 10200
~

S53 + Q54 + co54 - S54 = 7500
~

S63 + Q64 + co64 - S64 = 7350
~

S73 + Q74 + co74 - S74 = 1700
~

S83 + Q84 + co84 - S84 = 8712
~

S93 + Q94 + co94 - S94 = 4100
~

S103 + Q104 + co104 - S104 = 2170
~

Bulanık eşitlik halindeki kısıtlayıcılar modelde üçgensel üyelik fonksiyonları ile nitelendirilmiştir. Bu noktadan hareketle çalışma yapılan dönemlerde üretilmesi gereken yaklaşık miktarlar f_{nt} ile, tolerans miktarları p_{nt} ile, talep alt sınırı $f_{nt} - b_{nt}$ ve talep üst sınırı $b_{nt} + p_{nt}$ ile ifade edilmiştir. Ayrıca ilgili döneme ait toplam üretim miktarı X_{nt} ile ifade edilmektedir. Buna göre;

$$\mu(X_{nt}) = \begin{cases} 0, & X_{nt} \geq f_{nt} - p_{nt} \\ 1 - \dfrac{f_{nt} - X_{nt}}{p_{nt}}, & f_{nt} - p_{nt} < X_{nt} < f_{nt} \\ 1, & X_{nt} = f_{nt} \\ 1 - \dfrac{X_{nt} - f_{nt}}{p_{nt}}, & f_{nt} < X_{nt} < f_{nt} - p_{nt} \\ 0, & X_{nt} \geq f_{nt} + p_{nt} \end{cases}$$

Örneğin 1.ürünün 1.dönemi için üyelik fonksiyonu

$$\mu(X_{11}) = \begin{cases} 0, & X_{11} \geq 3000 - 400 \\ 1 - \dfrac{f_{11} - X_{11}}{p_{11}}, & 3000 - 400 < X_{11} < 3000 \\ 1, & X_{11} = 3000 \\ 1 - \dfrac{X_{11} - f_{11}}{p_{11}}, & 3000 < X_{11} < 3000 + 400 \\ 0, & X_{11} \geq 3000 + 400 \end{cases}$$

şeklinde yazılabilir.

Başlangıç stoğunun 0 olduğu varsayılarak

1.dönemin bulanık stok kısıtları

$0 + Q11 + co11 - S11 + 400\ \lambda \leq 3400$

$0 + Q11 + co11 - S11 - 400\ \lambda \geq 2600$

$0 + Q21 + co21 - S21 + 300\ \lambda \leq 2600$

$0 + Q21 + co21 - S21 - 300\ \lambda \geq 2000$

$0 + Q31 + co31 - S31 + 300\ \lambda \leq 2900$

$0 + Q31 + co31 – S31 - 300\ \lambda \geq 2300$

$0 + Q41 + co41 – S41 + 150\ \lambda \leq 2450$

$0 + Q41 + co41 – S41 - 150\ \lambda \geq 2150$

$0 + Q51 + co51 – S51 + 300\ \lambda \leq 3900$

$0 + Q51 + co51 – S51 - 300\ \lambda \geq 3300$

$0 + Q61 + co61 – S61 + 250\ \lambda \leq 4670$

$0 + Q61 + co61 – S61 - 250\ \lambda \geq 4170$

$0 + Q71 + co71 – S71 + 150\ \lambda \leq 5900$

$0 + Q71 + co71 – S71 - 150\ \lambda \geq 5500$

$0 + Q81 + co81 – S81 + 150\ \lambda \leq 2612$

$0 + Q81 + co81 – S81 - 150\ \lambda \geq 2252$

$0 + Q91 + co91 – S91 + 100\ \lambda \leq 1200$

$0 + Q91 + co91 – S91 - 100\ \lambda \geq 1000$

$0 + Q101 + co101 – S101 + 70\ \lambda \leq 560$

$0 + Q101 + co101 – S101 - 700\ \lambda \geq 420$

2.dönemin bulanık stok kısıtları

$S11 + Q12 + co12 - S12 + 400\,\lambda \leq 3600$

$S11 + Q12 + co12 - S12 - 400\,\lambda \geq 2800$

$S21 + Q22 + co22 - S22 + 300\,\lambda \leq 3300$

$S21 + Q22 + co22 - S22 - 300\,\lambda \geq 2700$

$S31 + Q32 + co32 - S32 + 300\,\lambda \leq 2700$

$S31 + Q32 + co32 - S32 - 300\,\lambda \geq 2100$

$S41 + Q42 + co42 - S42 + 150\,\lambda \leq 2850$

$S41 + Q42 + co42 - S42 - 150\,\lambda \geq 2550$

$S51 + Q52 + co52 - S52 + 300\,\lambda \leq 5100$

$S51 + Q52 + co52 - S52 - 300\,\lambda \geq 4500$

$S61 + Q62 + co62 - S62 + 250\,\lambda \leq 5290$

$S61 + Q62 + co62 - S62 - 250\,\lambda \geq 4790$

$S71 + Q72 + co72 - S72 + 400\,\lambda \leq 7600$

$S71 + Q72 + co72 - S72 - 400\,\lambda \geq 6800$

$S81 + Q82 + co82 - S82 + 180\,\lambda \leq 2628$

$S81 + Q82 + co82 - S82 - 180\,\lambda \geq 2268$

$S91 + Q92 + co92 - S92 + 100\,\lambda \leq 1000$

$S91 + Q92 + co92 - S92 - 100\,\lambda \geq 800$

$S101 + Q102 + co102 - S102 + 70\,\lambda \leq 560$

$S101 + Q102 + co102 - S102 - 70\,\lambda \geq 420$

3.dönemin bulanık stok kısıtları

$S12 + Q13 + co13 - S13 + 200\,\lambda \leq 1800$

$S12 + Q13 + co13 - S13 - 200\,\lambda \geq 1400$

$S22 + Q23 + co23 - S23 + 300\,\lambda \leq 1500$

$S22 + Q23 + co23 - S23 - 300\,\lambda \geq 900$

$S32 + Q33 + co33 - S33 + 300\,\lambda \leq 2700$

$S32 + Q33 + co33 - S33 - 300\,\lambda \geq 2100$

$S42 + Q43 + co43 - S43 + 150\,\lambda \leq 2250$

$S42 + Q43 + co43 - S43 - 150\,\lambda \geq 1950$

$S52 + Q53 + co53 - S53 + 300\,\lambda \leq 4800$

$S52 + Q53 + co53 - S53 - 300\,\lambda \geq 4200$

$S62 + Q63 + co63 - S63 + 250\,\lambda \leq 4870$

$S62 + Q63 + co63 - S63 - 250\,\lambda \geq 4370$

$S72 + Q73 + co73 - S73 + 200\,\lambda \leq 4900$

$S72 + Q73 + co73 - S73 - 200\,\lambda \geq 4500$

$S82 + Q83 + co83 - S83 + 180\,\lambda \leq 2484$

$S82 + Q83 + co83 - S83 - 180\,\lambda \geq 2124$

$S92 + Q93 + co93 - S93 + 100\,\lambda \leq 900$

$S92 + Q93 + co93 - S93 - 100\,\lambda \geq 700$

$S102 + Q103 + co103 - S103 + 140\,\lambda \leq 770$

$S102 + Q103 + co103 - S103 - 140\,\lambda \geq 490$

4.dönemin bulanık stok kısıtları

$S13 + Q14 + co14 - S14 + 200\,\lambda \geq 2600$

$S13 + Q14 + co14 - S14 - 200\,\lambda \leq 2200$

$S23 + Q24 + co24 - S24 + 300\,\lambda \geq 3600$

$S23 + Q24 + co24 - S24 - 300\,\lambda \leq 3000$

$S33 + Q34 + co34 - S34 + 450\,\lambda \geq 10350$

$S33 + Q34 + co34 - S34 - 450\ \lambda \leq 9450$

$S43 + Q44 + co44 - S44 + 600\ \lambda \geq 10800$

$S43 + Q44 + co44 - S44 - 600\ \lambda \leq 9600$

$S53 + Q54 + co54 - S54 + 450\ \lambda \geq 7950$

$S53 + Q54 + co54 - S54 - 450\ \lambda \leq 7050$

$S63 + Q64 + co64 - S64 + 500\ \lambda \geq 7850$

$S63 + Q64 + co64 - S64 - 500\ \lambda \leq 6850$

$S73 + Q74 + co74 - S74 + 100\ \lambda \geq 1800$

$S73 + Q74 + co74 - S74 - 100\ \lambda \leq 1600$

$S83 + Q84 + co84 - S84 + 360\ \lambda \geq 9072$

$S83 + Q84 + co84 - S84 - 360\ \lambda \leq 8352$

$S93 + Q94 + co94 - S94 + 200\ \lambda \geq 4300$

$S93 + Q94 + co94 - S94 - 200\ \lambda \leq 3900$

$S103 + Q104 + co104 - S104 + 140\ \lambda \geq 2310$

$S103 + Q104 + co104 - S104 - 140\ \lambda \leq 2030$

İşgücü kısıtı

$$\sum_{n=1}^{n} W_{nt-1}(Q_{nt-1} + co_{nt-1}) + H_t - u_{kt} - \sum_{n=1}^{n} W_{nt}(Q_{nt} + co_{nt}) = 0$$

Tablo 3.11. Dönemlere Ait Birim İşçilik Saatleri

Adam.dak/adet	1.Dönem	2.dönem	3.dönem	4.dönem
Ürün 1	1,95	2,05	1,93	1,92
Ürün 2	1,43	1,47	1,42	1,41
Ürün 3	1,62	1,69	1,61	1,61
Ürün 4	1,88	1,98	1,87	1,86
Ürün 5	1,28	1,31	1,28	1,28
Ürün 6	1,62	1,69	1,61	1,61
Ürün 7	2,71	2,89	2,68	2,65
Ürün 8	4,94	5,36	4,88	4,82
Ürün 9	3,22	3,46	3,19	3,16
Ürün 10	1,62	1,69	1,61	1,61

1.dönem işgücü seviyesi

H1 - F1 – 1,95Q11 – 1,95co11 – 1,43Q21 – 1,43co21 – 1,62Q31 – 1,62co31 – 1,88Q41 – 1,88co41 - 1.28Q51 – 1,28co51 – 1,62Q61 – 1,62co61 – 2,71Q71 – 2,71co71 – 4,94Q81 – 4,94co81 – 3,22Q91 – 3,22co91 – 1,62Q101 – 1,62co101 = 0

2.dönem işgücü seviyesi

1,95Q11 + 1,95co11 + 1,43Q21 + 1,43co 21 + 1,62Q31 + 1,62co 31 + 1,88Q41 + 1,88 co41 + 1,28Q51 + 1,28co51 + 1,62Q61 + 1,62co61 + 2,71Q71 + 2,71co71 + 4,94Q81 + 4,94co81 + 3,22Q91 + 3,22co91 + 1,62Q101 + 1,62co101 + H2 - F2 – 2,05Q12 - 2,05co12 – 1,47Q22 – 1,47co22 – 1,69Q32 – 1,69co32 - 1.98Q42 –

1,98co42 - 1.31Q52 - 1.31co52 – 1,69Q62 - 1.69co62 – 2,89Q72 – 2,89co72 – 5,36Q82 – 5,36co 82 – 3,46Q92 – 3,46co92 – 1,69Q102 – 1,69co102 = 0

3.dönem işgücü seviyesi

2,05Q12 + 2,05co12 + 1,47Q22 + 1,47co22 + 1,69Q32 + 1,69co32 + 1,98Q42 + 1,98co42 + 1,31Q52 + 1,31co52 + 1,69Q62 + 1,69co62 + 2,89Q72 + 2,89co72 + 5,36Q82 + 5,36co82 + 3,46Q92 + 3,46co 92 + 1,69Q102 + 1,69co102 + H3 - F3 – 1,93Q13 – 1,93co13 – 1,42Q23 – 1,42co23 – 1,61Q33 – 1,61co33 – 1,87Q43- 1.87co43 – 1,28Q53 – 1,28co53 – 1,61Q63 - 1,61co63 – 2,68Q73 – 2,68co73 – 4,88Q83 – 3,88co83 – 3,19Q93 – 3,19co93 – 1,61Q103 - 1.61co103 = 0 (5.128)

4.dönem işgücü seviyesi

1,93Q13 + 1,93co13 + 1,42Q23 + 1.42co23 + 1.61Q33 + 1.61co33 + 1.87Q43 + 1,87co43 + 1,28Q53 + 1,28co53 + 1,61Q63 + 1,61co63 + 2.68Q73 + 2.68co73 + 4,88Q83 + 4,88co83 + 3,19Q93 + 3,19co93 + 1,61Q103 + 1,61co103 + H4 - F4 – 1,92Q14 – 1,92co14 – 1,42Q24 – 1,42co24 – 1,61Q34 – 1,61co34 – 1,86Q44 – 1,86co44 – 1,28Q54 -1,28co54 – 1,61Q64 -1,61co64 – 2,65Q74 -2,65co74 – 4,82Q84 – 4,82co84 – 3,16Q94 – 3,16co94 – 1,61Q104 – 1,61co104 = 0

Burada önemli bir noktada dönemlere ait işgücü kapasiteleridir.

Aşağıda tablo 3.12 de bu dönemlere ait iş gücü kapasiteleri belirtilmiştir.

Tablo 3.12. İş gücü kapasiteleri

(adam.dak)	**1.dönem**	**2.dönem**	**3.dönem**	**4.dönem**
Teorik işçlik kapasitesi	376100	400400	374500	360000
Fiili işçilik	319700	376400	314500	299000
Fark	56400	24000	60000	61000

1.dönemin fiili iş gücü kısıtı

$$1{,}95Q11 + 1{,}95co11 + 1{,}43Q21 + 1{,}43co21 + 1{,}62Q31 + 1{,}62co31 + 1{,}88Q41 + 1{,}88co41 + 1.28Q51 + 1{,}28co51 + 1{,}62Q61 + 1{,}62co61 + 2{,}71Q71 + 2{,}71co71 + 4{,}94Q81 + 4{,}94co81 + 3{,}22Q91 + 3{,}22co91 + 1{,}62Q101 + 1{,}62co101 \lesssim 319700$$

2.dönemin fiili iş gücü kısıtı

$$2{,}05Q12 + 2{,}05co12 + 1{,}47Q22 + 1{,}47co22 + 1{,}69Q32 + 1{,}69co32 + 1.98Q42 + 1{,}98co42 + 1.31Q52 + 1.31co52 + 1{,}69Q62 + 1.69co62 + 2{,}89Q72 + 2{,}89co72 + 5{,}36Q82 + 5{,}36co\ 82 + 3{,}46Q92 + 3{,}46co92 + 1{,}69Q102 + 1{,}69co102 \lesssim 376400$$

3.dönemin fiili iş gücü kısıtı

$$1{,}93Q13 + 1{,}93co13 + 1{,}42Q23 + 1{,}42co23 + 1{,}61Q33 + 1{,}61co33 + 1{,}87Q43 + 1.87co43 + 1{,}28Q53 + 1{,}28co53 + 1{,}61Q63 + 1{,}61co63 + 2{,}68Q73 + 2{,}68co73 + 4{,}88Q83 + 3{,}88co83 + 3{,}19Q93 + 3{,}19co93 + 1{,}61Q103 + 1.61co103 \lesssim 314500$$

4.dönemin fiili iş gücü kısıtı

$$1{,}92Q14 + 1{,}92co14 + 1{,}42Q24 + 1{,}42co24 + 1{,}61Q34 + 1{,}61co34 + 1{,}86Q44 + 1{,}86co44 + 1{,}28Q54 + 1{,}28co54 + 1{,}61Q64 + 1{,}61co64 + 2{,}65Q74 + 2{,}65co74 + 4{,}82Q84 + 4{,}82co84 + 3{,}16Q94 + 3{,}16co94 + 1{,}61Q104 + 1{,}61co104 \lesssim 299000$$

Buna göre;

$$\sum_{n=1}^{n} W_{nt}(Q_{nt} + co_{nt}) \lesssim Wmax$$

Kısıtların sağ taraf sabitleri bulanık olarak tanımlanmıştır. Söz konusu kısıtlayıcıların sağ taraf sabitleri doğrusal olarak azalan üyelik fonksiyonları ile nitelenmiştir. Üyelik fonksiyonlarında tolerans p_{nt} olmak üzere, fiili süreler o_{nt} ile, teorik süreler ise

$o_{nt} + p_{nt}$ ile ifade edilmiştir. Dört dönem için işgücü kısıtına ait üyelik fonksiyonları Ax_1, Ax_2, Ax_3, Ax_4 tür ve aşağıdaki gibidir.

$$\mu(Ax_1) = \begin{cases} 0, & (\mathrm{Ax}_1) \geq 376100 \\ 1 - \dfrac{(\mathrm{Ax}_1) - 319700}{56400}, & 319500 < (\mathrm{Ax}_1) < 376100 \\ 1, & (\mathrm{Ax}_1) < 319500 \end{cases}$$

$$\mu(Ax_2) = \begin{cases} 0, & (\mathrm{Ax}_1) \geq 400400 \\ 1 - \dfrac{(\mathrm{Ax}_2) - 376400}{24000}, & 376400 < (\mathrm{Ax}_1) < 400400 \\ 1, & (\mathrm{Ax}_1) < 376400 \end{cases}$$

$$\mu(Ax_3) = \begin{cases} 0, & (\mathrm{Ax}_1) \geq 374500 \\ 1 - \dfrac{(\mathrm{Ax}_2) - 314500}{60000}, & 314500 < (\mathrm{Ax}_1) < 374500 \\ 1, & (\mathrm{Ax}_1) < 314500 \end{cases}$$

$$\mu(Ax_4) = \begin{cases} 0, & (\mathrm{Ax}_1) \geq 360000 \\ 1 - \dfrac{(\mathrm{Ax}_4) - 299000}{61000}, & 299000 < (\mathrm{Ax}_1) < 360000 \\ 1, & (\mathrm{Ax}_1) < 299000 \end{cases}$$

1.dönemin bulanıklaştırılmış filli iş gücü kısıtı

1,95Q11 + 1,95co11 + 1,43Q21 + 1,43co21 + 1,62Q31 + 1,62co31 + 1,88Q41 + 1,88co41 + 1.28Q51 + 1,28co51 + 1,62Q61 + 1,62co61 + 2,71Q71 + 2,71co71 + 4,94Q81 + 4,94co81 + 3,22Q91 + 3,22co91 + 1,62Q101 + 1,62co101 + 56400λ ≤376100

~

2.dönemin bulanıklaştırılmış filli iş gücü kısıtı

$$2{,}05Q12 + 2{,}05co12 + 1{,}47Q22 + 1{,}47co22 + 1{,}69Q32 + 1{,}69co32 + 1.98Q42 + 1{,}98co42 + 1.31Q52 + 1.31co52 + 1{,}69Q62 + 1.69co62 + 2{,}89Q72 + 2{,}89co72 + 5{,}36Q82 + 5{,}36co\ 82 + 3{,}46Q92 + 3{,}46co92 + 1{,}69Q102 + 1{,}69co102 + 24000\lambda \leq 400400$$

~

3.dönemin bulanıklaştırılmış filli iş gücü kısıtı

$$1{,}93Q13 + 1{,}93co13 + 1{,}42Q23 + 1{,}42co23 + 1{,}61Q33 + 1{,}61co33 + 1{,}87Q43 + 1.87co43 + 1{,}28Q53 + 1{,}28co53 + 1{,}61Q63 + 1{,}61co63 + 2{,}68Q73 + 2{,}68co73 + 4{,}88Q83 + 3{,}88co83 + 3{,}19Q93 + 3{,}19co93 + 1{,}61Q103 + 1.61co103 + 60000\lambda \leq 374500$$

~

4.dönemin bulanıklaştırılmış filli iş gücü kısıtı

$$1{,}92Q14 + 1{,}92co14 + 1{,}42Q24 + 1{,}42co24 + 1{,}61Q34 + 1{,}61co34 + 1{,}86Q44 + 1{,}86co44 + 1{,}28Q54 + 1{,}28co54 + 1{,}61Q64 + 1{,}61co64 + 2{,}65Q74 + 2{,}65co74 + 4{,}82Q84 + 4{,}82co84 + 3{,}16Q94 + 3{,}16co94 + 1{,}61Q104 + 1{,}61co104 \leq 360000$$

~

Amaç fonksiyonunun oluşturulması

Aşağıdaki verilen tablolardaki (Tablo 3.13. , Tablo 3.14. , Tablo 3.15. , Tablo 3.16.) değerlerden hareketle katsayılar belirlenir.

Katsayılarının belirlenmesi

Tablo 3.13. Normal Mesai Birim Maliyetleri (L_{nt})

(TL / adet)	**1.Dönem**	**2.Dönem**	**3.Dönem**	**4.Dönem**
Ürün1	1,63	1,33	2,42	1,93
Ürün2	1,71	1,97	2,14	1,86
Ürün3	1,15	1,91	1,39	1,32
Ürün4	2,31	2,25	2,33	2,41

Ürün5	2,22	1,91	3,12	2,27
Ürün6	32,33	25,23	30,47	41,08
Ürün7	1,67	1,34	1,43	1,84
Ürün8	1,58	1,24	2,27	2,13
Ürün9	2,15	1,84	2,25	2,37
Ürün10	2,05	1,81	2,17	2,31

Tablo 3.14. Fazla Mesai Birim Maliyetleri (o_{nt})

(TL / adet)	**1.Dönem**	**2.Dönem**	**3.Dönem**	**4.Dönem**
Ürün1	2,36	1,94	3,61	2,66
Ürün2	2,68	2,96	3,59	2,86
Ürün3	1,84	2,95	2,18	1,81
Ürün4	3,31	3,09	3,23	3,32
Ürün5	3,22	2,73	4,86	3,34
Ürün6	47,14	20,23	25,47	31,08
Ürün7	2,25	1,86	2,04	2,76
Ürün8	2,86	2,45	3,37	3,20
Ürün9	3,25	2,78	3,21	3,53
Ürün10	3,05	2,64	3,24	3,58

Tablo 3.15. Stok Taşıma Birim Maliyetleri (h_{nt})

(TL / adet)	**1.Dönem**	**2.Dönem**	**3.Dönem**	**4.Dönem**
Ürün1	1.96	1.61	3.00	2.21
Ürün2	2.23	2.46	2.99	2.38
Ürün3	1.53	2.45	1.81	1.50
Ürün4	2.75	2.57	2.69	2.76
Ürün5	2.68	2.27	4.04	2.78
Ürün6	39.26	16.85	21.21	25.88
Ürün7	1.87	1.54	1.69	2.29
Ürün8	2.38	2.04	2.80	2.66
Ürün9	2.70	2.31	2.67	2.94
Ürün10	2.54	2.19	2.69	2.98

Tablo 3.16. İşe Alma ve İşten Çıkarma Birim Maliyetleri

(TL/adam.dak)	**1.Dönem**	**2.Dönem**	**3.Dönem**	**4.Dönem**
İşe alma k_t	3,85	3,40	3,25	3,36
İşten çıkarma u_t	0,75	0,75	0,75	0,75

Modelin amaç fonksiyonu birçok bütünleşik üretim planlama modellerinde olduğu gibi maliyetlerin minimizasyonu şeklindedir. Toplam maliyet; 4 dönemlik periyot süresince karşılaşılan üretim maliyetlerinin ve işgücü seviyesindeki değişikliğin neden olduğu maliyetlerin toplamıdır. Bu noktadan hareketle amaç fonksiyonu şu şekildedir.

$$\text{Zmin} = \sum_{n=1}^{N}\sum_{t=1}^{T}(L_{nt}.Q_{nt} + o_{nt}.co_{nt} + h_{nt}.S_{nt}) + \sum_{t=1}^{T}(k_t.H_t + u_t.uk_t)$$

$\sum_{n=1}^{N}\sum_{t=1}^{T} L_{nt}.Q_{nt}$: İşçilik Maliyeti

$\sum_{n=1}^{N}\sum_{t=1}^{T} o_{nt}.co_{nt}$: Fazla Mesai Maliyeti

$\sum_{n=1}^{N}\sum_{t=1}^{T} h_{nt}.S_{nt}$: Stok Taşıma Maliyeti

$\sum_{t=1}^{T} k_t.H_t$: İşe alma maliyeti

$\sum_{t=1}^{T} u_t.uk_t$: İşten çıkarma maliyeti

$$Q_{nt}, S_{nt}, L_{nt}H_t.co_{nt}, o_{nt}, h_{nt}, k_t, u_t, uk_t \geq 0$$

Katsayılar yerine yazıldığında amaç fonksiyonu aşağıdaki gibi elde edilir.

Zmin = 1,63Q11 + 1.71Q21 + 1,15Q31 + 2,31Q41 + 2,22Q51 + 32,33Q61 + 1,67Q71 + 1,58Q81 + 2,15Q91 + 2,05Q101 + 1,33Q12 + 1,97Q22 + 1,91Q32 + 2,25Q42 + 1,91Q52 + 20,23Q62 + 1,34Q72 + 1,24 Q82 + 1.84Q92 + 1,81Q102 + 2,42Q13 + 2,14Q23 + 1,39Q33 + 2,33Q43 + 3,12 Q53 + 25,47Q63 + 1,43Q73 + 2,27Q83 + 2,25Q93 + 2,17Q103 + 1,93Q14 + 1,86Q24 + 1,32Q34 + 2,41Q44 + 2,27 Q54 + 31,08Q64 + 1,84Q74 + 2,13Q84 + 2,37Q94 + 2,31Q104 + 2,36co11 + 2,68co21 + 1,84co31 + 3,31co41 + 3,22co51 + 47,14co61 + 2,25co71 + 2,86co81 + 3,25co91 + 3,05 co Y101 + 1,94co12 + 2,96co22 + 2,95co32 + 3,09co42 + 2,73co52 + 25,23co62 + 1,86co72 + 2.45co82 + 2,78co92 + 2,64co102 + 3,61co13 + 3,59co23 + 2,18co33 + 3,23co43 + 4,86co53 + 30,47co63 + 2,04co73 + 3,37co83 + 3,21co93 + 3,24co103 + 2,66coY14 + 2,86co24 + 1,81co34 + 3,32co44 + 3,34co54 + 41,08co64 + 2,76co74 + 3,20co84 + 3,53co94 + 3,58co104 + 1,96S11 + 2,23S21 + 1,53S31 + 2,75S41 + 2,68S51 + 39,26 S61 + 1,87S71 + 2,38S81 + 2,70S91 + 2,54S101 + 1,61S12 + 2,46S22 + 2,45S32 + 2,57S42 + 2,27S52 + 20,85S62 + 1,54S72 + 2,04S82 +2,31S92 + 2,19S102 + 3,00S13 + 2,99S23 + 1,81S33 + 2,69S43+ 4,04S53 + 26,21S63 + 1.69S73 + 2,80S83 + 2,67S93 + 2,69S103 + 2,21S14 + 2,38S24 + 1,50S34 + 2,76S44 + 2,78S54 + 35,31S64 + 2,29S74 + 2.66S84 + 2,94S94 + 2,98S104 + 3,85H1 + 0,7 uk1 + 3,40H2 + 0,75 uk2 + 3,25 H3 + 0,75uk3 + 3,36H4 + 0,75uk4

Amaç fonksiyonundaki bulanıklık, sübjektif olarak belirlenebilen erişim düzeyi (b_{nt}) ve bu erişim düzeyine ilişkin tolerans miktarı (p_{nt}) ile belirtilmektedir. 4 dönemlik planlama için 1175000 TL civarı veya daha az şeklinde bir üretim maliyetini hedeflendiği ayrıca belirlenen 4 dönemlik erişim düzeyi için tanımlanan tolerans miktarı ise 75000TL olarak kabul edilmiştir. Bulanık amaç fonksiyonu, bulanık eşitsizlik kısıtlayıcılarına benzer şekilde, parçalı doğrusal üyelik fonksiyonu ile nitelenmiştir. Buna göre, amaç fonksiyonuna ilişkin üyelik fonksiyonu aşağıda verildiği gibidir.

$$\mu(Zx) = \begin{cases} 0, & c^{T}x \geq 1250000 \\ 1 - \dfrac{c^{T}x - 1200000}{100000}, & 1175000 < (c^{T}x) < 1250000 \\ 1, & c^{T}x \leq 1175000 \end{cases}$$

Bulanık amaç ve bulanık kısıtlayıcıların karşılanma derecesi λ ile gösterilmektedir. Bulanık karar kümesinin en yüksek üyelik dereceli elemanı (veya λ'yı) belirleyebilmek için aşağıda verilen doğrusal optimizasyon probleminin çözülmesi gerekir. Bu noktadan hareketle:

λ maks

1,63Q11 + 1.71Q21 + 1,15Q31 + 2,31Q41 + 2,22Q51 + 32,33Q61 + 1,67Q71 + 1,58Q81 + 2,15Q91 + 2,05Q101 + 1,33Q12 + 1,97Q22 + 1,91Q32 + 2,25Q42 + 1,91Q52 + 20,23Q62 + 1,34Q72 + 1,24 Q82 + 1.84Q92 + 1,81Q102 + 2,42Q13 + 2,14Q23 + 1,39Q33 + 2,33Q43 + 3,12 Q53 + 25,47Q63 + 1,43Q73 + 2,27Q83 + 2,25Q93 + 2,17Q103 + 1,93Q14 + 1,86Q24 + 1,32Q34 + 2,41Q44 + 2,27 Q54 + 31,08Q64 + 1,84Q74 + 2,13Q84 + 2,37Q94 + 2,31Q104 + 2,36co11 + 2,68co21 + 1,84co31 + 3,31co41 + 3,22co51 + 47,14co61 + 2,25co71 + 2,86co81 + 3,25co91 + 3,05 co101 + 1,94co12 + 2,96co22 + 2,95co32 + 3,09co42 + 2,73co52 + 25,23co62 + 1,86co72 + 2.45co82 + 2,78co92 + 2,64co102 + 3,61co13 + 3,59co23 + 2,18co33 + 3,23co43 + 4,86co53 + 30,47co63 + 2,04co73 + 3,37co83 + 3,21co93 + 3,24co103 + 2,66co14 + 2,86co24 + 1,81co34 + 3,32co44 + 3,34co54 + 41,08co64 + 2,76co74 + 3,20co84 + 3,53co94 + 3,58co104 + 1,96S11 + 2,23S21 + 1,53S31 + 2,75S41 + 2,68S51 + 39,26 S61 + 1,87S71 + 2,38S81 + 2,70S91 + 2,54S101 + 1,61S12 + 2,46S22 + 2,45S32 + 2,57S42 + 2,27S52 + 20,85S62 + 1,54S72 + 2,04S82 +2,31S92 + 2,19S102 + 3,00S13 + 2,99S23 + 1,81S33 + 2,69S43+ 4,04S53 + 26,21S63 + 1.69S73 + 2,80S83 + 2,67S93 + 2,69S103 + 2,21S14 + 2,38S24 + 1,50S34 + 2,76S44 + 2,78S54 + 35,31S64 + 2,29S74 + 2.66S84 + 2,94S94 + 2,98S104 + 3,85H1 + 0,7 uk1 + 3,40H2 + 0,75 uk2 + 3,25 H3 + 0,75uk3 + 3,36H4 + 0,75uk4 +100000 λ≤ 1300000

2. $Q11 + co11 - S11 + 400\ \lambda \leq 3400$
3. $Q11 + co11 - S11 - 400\ \lambda \geq 2600$
4. $Q21 + co21 – S21 + 300\ \lambda \leq 2600$
5. $Q21 + co21 – S21 - 300\ \lambda \geq 2000$
6. $Q31 + co31 – S31 + 300\ \lambda \leq 2900$
7. $Q31 + co31 – S31 - 300\ \lambda \geq 2300$
8. $Q41 + co41 – S41 + 150\ \lambda \leq 2450$
9. $Q41 + co41 – S41 - 150\ \lambda \geq 2150$
10. $Q51 + co51 – S51 + 300\ \lambda \leq 3900$
11. $Q51 + co51 – S51 - 300\ \lambda \geq 3300$
12. $Q61 + co61 – S61 + 250\ \lambda \leq 4670$
13. $Q61 + co61 – S61 - 250\ \lambda \leq 4670$
14. $Q71 + co71 – S71 + 150\ \lambda \leq 5900$
15. $Q71 + co71 – S71 - 150\ \lambda \geq 5500$
16. $Q81 + co81 – S81 + 150\ \lambda \leq 2612$
17. $Q81 + co81 – S81 - 150\ \lambda \geq 2252$
18. $Q91 + co91 – S91 + 100\ \lambda \leq 1200$
19. $Q91 + co91 – S91 - 100\ \lambda \geq 1000$
20. $Q101 + co101 – S101 + 70\ \lambda \leq 560$
21. $Q101 + co101 – S101 - 700\ \lambda \geq 420$
22. $S11 + Q12 + co12 - S12 + 400\ \lambda \leq 3600$
23. $S11 + Q12 + co12 - S12 - 400\ \lambda \geq 2800$
24. $S21 + Q22 + co22 - S22 + 300\ \lambda \leq 3300$
25. $S21 + Q22 + co22 - S22 - 300\ \lambda \geq 2700$
26. $S31 + Q32 + co32 - S32 + 300\ \lambda \leq 2700$
27. $S31 + Q32 + co32 - S32 - 300\ \lambda \geq 2100$
28. $S41 + Q42 + co42- S42 + 150\ \lambda \leq 2850$
29. $S41 + Q42 + co42- S42 - 150\ \lambda \geq 2550$
30. $S51 + Q52 + co52 - S52 + 300\ \lambda \leq 5100$
31. $S51 + Q52 + co52 - S52 - 300\ \lambda \geq 4500$
32. $S61 + Q62 + co62 - S62 + 250\ \lambda \leq 5290$

33. $S61 + Q62 + co62 - S62 - 250\lambda \geq 4790$
34. $S71 + Q72 + co72 - S72 + 400\lambda \leq 7600$
35. $S71 + Q72 + co72 - S72 - 400\lambda \geq 6800$
36. $S81 + Q82 + co82 - S82 + 180\lambda \leq 2628$
37. $S81 + Q82 + co82 - S82 - 180\lambda \geq 2268$
38. $S91 + Q92 + co92 - S92 + 100\lambda \leq 1000$
39. $S91 + Q92 + co92 - S92 - 100\lambda \geq 800$
40. $S101 + Q102 + co102 - S102 + 70\lambda \leq 560$
41. $S101 + Q102 + co102 - S102 - 70\lambda \geq 420$
42. $S12 + Q13 + co13 - S13 + 200\lambda \leq 1800$
43. $S12 + Q13 + co13 - S13 - 200\lambda \geq 1400$
44. $S22 + Q23 + co23 - S23 + 300\lambda \leq 1500$
45. $S22 + Q23 + co23 - S23 - 300\lambda \geq 900$
46. $S32 + Q33 + co33 - S33 + 300\lambda \leq 2700$
47. $S32 + Q33 + co33 - S33 - 300\lambda \geq 2100$
48. $S42 + Q43 + co43 - S43 + 150\lambda \leq 2250$
49. $S42 + Q43 + co43 - S43 - 150\lambda \geq 1950$
50. $S52 + Q53 + co53 - S53 + 300\lambda \leq 4800$
51. $S52 + Q53 + co53 - S53 - 300\lambda \geq 4200$
52. $S62 + Q63 + co63 - S63 + 250\lambda \leq 4870$
53. $S62 + Q63 + co63 - S63 - 250\lambda \geq 4370$
54. $S72 + Q73 + co73 - S73 + 200\lambda \leq 4900$
55. $S72 + Q73 + co73 - S73 - 200\lambda \geq 4500$
56. $S82 + Q83 + co83 - S83 + 180\lambda \leq 2484$
57. $S82 + Q83 + co83 - S83 - 180\lambda \geq 2124$
58. $S92 + Q93 + co93 - S93 + 100\lambda \leq 900$
59. $S92 + Q93 + co93 - S93 - 100\lambda \geq 700$
60. $S102 + Q103 + co103 - S103 + 140\lambda \leq 770$
61. $S102 + Q103 + co103 - S103 - 140\lambda \geq 490$
62. $S13 + Q14 + co14 - S14 + 200\lambda \geq 2600$
63. $S13 + Q14 + co14 - S14 - 200\lambda \leq 2200$
64. $S23 + Q24 + co24 - S24 + 300\lambda \geq 3600$

65. $S23 + Q24 + co24 - S24 - 300\lambda \leq 3000$

66. $S33 + Q34 + co34 - S34 + 450\lambda \geq 10350$

67. $S33 + Q34 + co34 - S34 - 450\lambda \leq 9450$

68. $S43 + Q44 + co44 - S44 + 600\lambda \geq 10800$

69. $S43 + Q44 + co44 - S44 - 600\lambda \leq 9600$

70. $S53 + Q54 + co54 - S54 + 450\lambda \geq 7950$

71. $S53 + Q54 + co54 - S54 - 450\lambda \leq 7050$

72. $S63 + Q64 + co64 - S64 + 500\lambda \geq 7850$

73. $S63 + Q64 + co64 - S64 - 500\lambda \leq 6850$

74. $S73 + Q74 + co74 - S74 + 100\lambda \geq 1800$

75. $S73 + Q74 + co74 - S74 - 100\lambda \leq 1600$

76. $S83 + Q84 + co84 - S84 + 360\lambda \geq 9072$

77. $S83 + Q84 + co84 - S84 - 360\lambda \leq 8352$

78. $S93 + Q94 + co94 - S94 + 200\lambda \geq 4300$

79. $S93 + Q94 + co94 - S94 - 200\lambda \leq 3900$

80. $S103 + Q104 + co104 - S104 + 140\lambda \geq 2310$

81. $S103 + Q104 + co104 - S104 - 140\lambda \leq 2030$

82. $H1 - uk1 - 1{,}95Q11 - 1{,}95co11 - 1{,}43Q21 - 1{,}43co21 - 1{,}62Q31 - 1{,}62co31 - 1{,}88Q41 - 1{,}88co41 - 1.28Q51 - 1{,}28co51 - 1{,}62Q61 - 1{,}62co61 - 2{,}71Q71 - 2{,}71co71 - 4{,}94Q81 - 4{,}94co81 - 3{,}22Q91 - 3{,}22co91 - 1{,}62Q101 - 1{,}62co101 = 0$

83. $1{,}95Q11 + 1{,}95co11 + 1{,}43Q21 + 1{,}43co\ 21 + 1{,}62Q31 + 1{,}62co\ 31 + 1{,}88Q41 + 1{,}88\ co41 + 1{,}28Q51 + 1{,}28co51 + 1{,}62Q61 + 1{,}62co61 + 2{,}71Q71 + 2{,}71co71 + 4{,}94Q81 + 4{,}94co81 + 3{,}22Q91 + 3{,}22co91 + 1{,}62Q101 + 1{,}62co101 + H2 - uk2 - 2{,}05Q12 - 2{,}05co12 - 1{,}47Q22 - 1{,}47co22 - 1{,}69Q32 - 1{,}69co32 - 1.98Q42 - 1{,}98co42 - 1.31Q52 - 1.31co52 - 1{,}69Q62 - 1.69co62 - 2{,}89Q72 - 2{,}89co72 - 5{,}36Q82 - 5{,}36co\ 82 - 3{,}46Q92 - 3{,}46co92 - 1{,}69Q102 - 1{,}69co102 = 0$

84. $2{,}05Q12 + 2{,}05co12 + 1{,}47Q22 + 1{,}47co22 + 1{,}69Q32 + 1{,}69co32 + 1{,}98Q42 + 1{,}98co42 + 1{,}31Q52 + 1{,}31co52 + 1{,}69Q62 + 1{,}69co62 + 2{,}89Q72 + 2{,}89co72 + 5{,}36Q82 + 5{,}36co82 + 3{,}46Q92 + 3{,}46co\ 92 + 1{,}69Q102 + 1{,}69co102 +$

H3 - uk3 – 1,93Q13 – 1,93co13 – 1,42Q23 – 1,42co23 – 1,61Q33 – 1,61co33 – 1,87Q43- 1.87co43 – 1,28Q53 – 1,28co53 – 1,61Q63 - 1,61co63 – 2,68Q73 – 2,68co73 – 4,88Q83 – 3,88co83 – 3,19Q93 – 3,19co93 – 1,61Q103 - 1.61co103 = 0

85. 1,93Q13 + 1,93co13 + 1,42Q23 + 1.42co23 + 1.61Q33 + 1.61co33 + 1.87Q43 + 1,87co43 + 1,28Q53 + 1,28co53 + 1,61Q63 + 1,61co63 + 2.68Q73 + 2.68co73 + 4,88Q83 + 4,88co83 + 3,19Q93 + 3,19co93 + 1,61Q103 + 1,61co103 + H4 – uk4 – 1,92Q14 – 1,92co14 – 1,42Q24 – 1,42co24 – 1,61Q34 – 1,61co34 – 1,86Q44 – 1,86co44 – 1,28Q54 -1,28co54 – 1,61Q64 -1,61co64 – 2,65Q74 - 2,65co74 – 4,82Q84 – 4,82co84 – 3,16Q94 – 3,16co94 – 1,61Q104 – 1,61co104 = 0

86. 1,95Q11 + 1,95co11 + 1,43Q21 + 1,43co21 + 1,62Q31 + 1,62co31 + 1,88Q41 + 1,88co41 + 1.28Q51 + 1,28co51 + 1,62Q61 + 1,62co61 + 2,71Q71 + 2,71co71 + 4,94Q81 + 4,94co81 + 3,22Q91 + 3,22co91 + 1,62Q101 + 1,62co101 + 56400λ ≤376100

87. 2,05Q12 + 2,05co12 + 1,47Q22 + 1,47co22 + 1,69Q32 + 1,69co32 + 1.98Q42 + 1,98co42 + 1.31Q52 + 1.31co52 + 1,69Q62 + 1.69co62 + 2,89Q72 + 2,89co72 + 5,36Q82 + 5,36co 82 + 3,46Q92 + 3,46co92 + 1,69Q102 + 1,69co102 + 24000λ ≤ 400400

88. 1,93Q13 + 1,93co13 + 1,42Q23 + 1,42co23 + 1,61Q33 + 1,61co33 + 1,87Q43 + 1.87co43 + 1,28Q53 + 1,28co53 + 1,61Q63 + 1,61co63 + 2,68Q73 + 2,68co73 + 4,88Q83 + 3,88co83 + 3,19Q93 + 3,19co93 + 1,61Q103 + 1.61co103 + 60000λ ≤ 374500

89. 1,92Q14 + 1,92co14 + 1,42Q24 + 1,42co24 + 1,61Q34 + 1,61co34 + 1,86Q44 + 1,86co44 + 1,28Q54 + 1,28co54 + 1,61Q64 + 1,61co64 + 2,65Q74 + 2,65co74 + 4,82Q84 + 4,82co84 + 3,16Q94 + 3,16co94 + 1,61Q104 + 1,61co104 + 61000λ ≤ 360000

Kısıtlar ve amaç fonksiyonu belirtildikten sonra Lindo programında sonuçlar aşağıdaki gibi bulunmuştur.

Tablo 3.17. Bulanık Optimizasyon Sonuçlar

λ (x) = 0.156216					
Variable	**Value**	**Reduce Cost**	**Variable**	**Value**	**Reduce Cost**
Q11	2813.846210	0.000000	Q83	2328.286715	0.000000
Q21	2142.802115	0.000000	Q93	4837.674125	0.000000
Q31	2153.402109	0.000000	Q103	746.656293	0.000000
Q41	2224.501142	0.000000	Q14	323.925171	0.000000
Q51	3318.713116	0.000000	Q24	3492.125117	0.000000
Q61	3952.133054	0.000000	Q34	9549.281892	0.000000
Q71	4947.025124	0.000000	Q44	9688.744927	0.000000
Q81	3285.608154	0.000000	Q54	7287.684341	0.000000
Q91	1025.743641	0.000000	Q64	7047.970012	0.000000
Q101	451.381874	0.000000	Q74	0.000000	0.000030
Q12	2123.576242	0.000000	Q84	7413.526886	0.000000
Q22	2694.922119	0.000000	Q94	0.000000	0.000031
Q32	2192.815194	0.000000	Q104	2182.608318	0.000000
Q42	2635.560132	0.000000	co11	0.000000	0.000002
Q52	4395.722563	0.000000	co21	0.000000	0.000005
Q62	4063.835716	0.000000	co31	0.000000	0.000004
Q72	6742.821156	0.000000	co41	0.000000	0.000007
Q82	1437.997628	0.000000	co51	0.000000	0.000002
Q92	829.964154	0.000000	co61	0.000000	0.000072
Q102	443.218753	0.000000	co71	0.000000	0.000003
Q13	3479.874816	0.000000	co81	0.000000	0.000004
Q23	1025.232172	0.000000	co91	0.000000	0.000003
Q33	2489.679883	0.000000	co101	0.000000	0.000003
Q43	2213.739876	0.000000	co12	0.000000	0.000004
Q53	4198.929874	0.000000	co22	0.000000	0.000005
Q63	3802.425702	0.000000	co32	0.000000	0.000003
Q73	5489.125071	0.000000	co42	0.000000	0.000003
co52	0.000000	0.000004	S11	0.000000	0.000005
co62	0.000000	0.000004	S21	0.000000	0.000004

Tablo 3.17. Devam

co72	0.000000	0.000005	S31	0.000000	0.000005
co82	0.000000	0.000003	S41	0.000000	0.000008
co92	0.000000	0.000003	S51	0.000000	0.000001
co102	0.000000	0.000005	S61	0.000000	0.000117
co13	0.000000	0.000005	S71	385.218604	0.000000
co23	0.000000	0.000003	S32	0.000000	0.000032
co33	0.000000	0.000005	S42	0.000000	0.000017
co43	0.000000	0.000003	S52	0.000000	0.000063
co53	0.000000	0.000005	S62	0.000000	0.000061
co63	0.000000	0.000007	S72	0.000000	0.000024
co73	0.000000	0.000003	S82	0.000000	0.000053
co83	0.000000	0.000005	S82	0.000000	0.000053
co93	0.000000	0.000005	S92	0.000000	0.000044
co103	0.000000	0.000005	S102	0.000000	0.000021
co14	0.000000	0.000004	S13	1834.043518	0.000000
co24	0.000000	0.000004	S23	0.000000	0.000001
co34	0.000000	0.000003	S33	0.000000	0.000002
co44	0.000000	0.000005	S43	0.000000	0.000006
co54	0.000000	0.000005	S53	0.000000	0.000003
co64	0.000000	0.000007	S63	0.000000	0.000134
co74	0.000000	0.000003	S73	1642.941216	0.000000
co84	0.000000	0.000004	S83	0.000000	0.000064
co94	0.000000	0.000003	S93	0.000000	0.000000
co104	0.000000	0.000004	S104	0.000000	0,000041

H1	63185.368	0.000000
uk1	0.000000	0.000019
H2	0.000000	0.000014
uk2	0.000000	0.000009
H3	28023.309	0.000000
uk3	0.000000	0.000012
H4	0.000000	0.000000
uk4	0.000000	0.000024

1.175.000TL'lik maliyet hedefine 75.000 TL lik tolerans miktarı ile çözümlendiğinde, [1.100000 , 1.250.000] aralığında bulanık olarak ifade edilen amaç fonksiyonu, çözüm değerlerine göre 1.167.184 TL olarak bulunur. λ = 0,156216 bulunarak, gösterilen tolerans miktarı; $0.156216.(75000) = 11.716{,}200$ TL olarak bulunmuştur.

B) Doğrusal optimizasyonla çözüm

Model bulanıklaştırılmadan amaç fonksiyonu ve kısıtlar aşağıdaki gibi yazılarak sonuç elde edilir.

Amaç fonksiyonu

Zmin = 1,63Q11 + 1.71Q21 + 1,15Q31 + 2,31Q41 + 2,22Q51 + 32,33Q61 + 1,67Q71 + 1,58Q81 + 2,15Q91 + 2,05Q101 + 1,33Q12 + 1,97Q22 + 1,91Q32 + 2,25Q42 + 1,91Q52 + 25,32Q62 + 1,34Q72 + 1,24 Q82 + 1.84Q92 + 1,81Q102 + 2,42Q13 + 2,14Q23 + 1,39Q33 + 2,33Q43 + 3,12 Q53 + 29,87Q63 + 1,43Q73 + 2,27Q83 + 2,25Q93 + 2,17Q103 + 1,93Q14 + 1,86Q24 + 1,32Q34 + 2,41Q44 + 2,27 Q54 + 31,08Q64 + 1,84Q74 + 2,13Q84 + 2,37Q94 + 2,31Q104 + 2,36co11 + 2,68co21 + 1,84co31 + 3,31co41 + 3,22co51 + 47,14co61 + 2,25co71 + 2,86co81 + 3,25co91 + 3,05 co101 + 1,94co12 + 2,96co22 + 2,95co32 + 3,09co42 + 2,73co52 + 25,23co62 + 1,86co72 + 2.45co82 + 2,78co92 + 2,64co102 + 3,61co13 + 3,59co23 + 2,18co33 + 3,23co43 + 4,86co53 + 30,47co63 + 2,04co73 + 3,37co83 + 3,21co93 + 3,24co103 + 2,66co14 + 2,86co24 + 1,81co34 + 3,32co44 + 3,34co54 + 41,08co64 + 2,76co74 + 3,20co84 + 3,53co94 + 3,58co104 + 1,96S11 + 2,23S21 + 1,53S31 + 2,75S41 + 2,68S51 + 39,26 S61 + 1,87S71 + 2,38S81 + 2,70S91 + 2,54S101 + 1,61S12 + 2,46S22 + 2,45S32 + 2,57S42 + 2,27S52 + 20,85S62 + 1,54S72 + 2,04S82 +2,31S92 + 2,19S102 + 3,00S13 + 2,99S23 + 1,81S33 + 2,69S43+ 4,04S53 + 26,21S63 + 1.69S73 + 2,80S83 + 2,67S93 + 2,69S103 + 2,21S14 + 2,38S24 + 1,50S34 + 2,76S44 + 2,78S54 + 35,31S64 + 2,29S74 + 2.66S84 + 2,94S94 + 2,98S104 + 3,85H1 + 0,7 uk1 + 3,40H2 + 0,75 uk2 + 3,25 H3 + 0,75uk3 + 3,36H4 + 0,75uk4

2. Q11 + co11 – S11 = 3000
3. Q21 + co21 – S21 = 2300
4. Q31 + co31 – S31 = 2600
5. Q41 + co41 – S41 = 2300
6. Q51 + co51 – S51 = 3600
7. Q61 + co61 – S61 =4420
8. Q71 + co71 – S71 = 5700
9. Q81 + co81 – S81 = 2432
10. Q91 + co91 – S91 =1100
11. Q101 + co101 – S101 = 490
12. S11 + Q12 + co12 – S12 = 3200
13. S21 + Q22 + co22 – S22 = 3000
14. S31 + Q32 + co32 – S32 = 2400
15. S41 + Q42 + co42 – S42 = 2700
16. S51 + Q52 + co52 – S52 = 4800
17. S61 + Q62 + co62 – S62 = 5040
18. S71 + Q72 + co72 – S72 = 7200
19. S81 + Q82 + co82 – S82 = 2448
20. S91 + Q92 + co92 – S92= 900
21. S101 + Q102 + co102 – S102 = 490
22. S12 + Q13 + co13 – S13 = 1600
23. S22 + Q23 + co23 – S23 = 1200
24. S32 + Q33 + co33 – S33 = 2400
25. S42 + Q43 + co43 – S43 = 2100
26. S52 + Q53 + co53 – S53 = 4500
27. S62 + Q63 + co63 – S63 = 4620
28. S72 + Q73 + co73 – S73 = 4700
29. S82 + Q83 + co83 – S83 = 2304
30. S92 + Q93 + co93 – S93 = 800
31. S102 + Q103 + co103 – S103 = 630
32. S13 + Q14 + co14 – S14 = 2400
33. S23 + Q24 + co24 – S24 = 3300
34. S33 + Q34 + co34 – S34 = 9900

35. S43 + Q44 + co44 – S44 = 10200

36. S53 + Q54 + co54 – S54 =7500

37. S63 + Q64 + co64 – S64 = 7350

38. S73 + Q74 + co74 – S74 = 1700

39. S83 + Q84 + co84 – S84 = 8712

40. S93 + Q94 + co94 – S94 = 4100

41. S103 + Q104 + co104 – S104 = 2170

42. H1 - uk1 – 1,95Q11 – 1,95co11 – 1,43Q21 – 1,43co21 – 1,62Q31 – 1,62co31-1,88Q41 – 1,88co41 - 1.28Q51 – 1,28co51 – 1,62Q61 – 1,62co61 – 2,71Q71 – 2,71co71 – 4,94Q81 – 4,94co81 – 3,22Q91 – 3,22co91 – 1,62Q101 – 1,62co101=0

43. 1,95Q11 + 1,95co11 + 1,43Q21 + 1,43co 21 + 1,62Q31 + 1,62co 31 + 1,88Q41 + 1,88 co41 + 1,28Q51 + 1,28co51 + 1,62Q61 + 1,62co61 + 2,71Q71 + 2,71co71 + 4,94Q81 + 4,94co81 + 3,22Q91 + 3,22co91 + 1,62Q101 + 1,62co101 + H2 - uk2 – 2,05Q12 - 2,05co12 – 1,47Q22 – 1,47co22 – 1,69Q32 – 1,69co32 - 1.98Q42 – 1,98co42 - 1.31Q52 - 1.31co52 – 1,69Q62 - 1.69co62 – 2,89Q72 – 2,89co72 – 5,36Q82 – 5,36co 82 – 3,46Q92 – 3,46co92 – 1,69Q102 – 1,69co102 = 0

44. 2,05Q12 + 2,05co12 + 1,47Q22 + 1,47co22 + 1,69Q32 + 1,69co32 + 1,98Q42 + 1,98co42 + 1,31Q52 + 1,31co52 + 1,69Q62 + 1,69co62 + 2,89Q72 + 2,89co72 + 5,36Q82 + 5,36co82 + 3,46Q92 + 3,46co 92 + 1,69Q102 + 1,69co102 + H3 - uk3 – 1,93Q13 – 1,93co13 – 1,42Q23 – 1,42co23 – 1,61Q33 – 1,61co33 – 1,87Q43- 1.87co43 – 1,28Q53 – 1,28co53 – 1,61Q63 - 1,61co63 – 2,68Q73 – 2,68co73 – 4,88Q83 – 3,88co83 – 3,19Q93 – 3,19co93 – 1,61Q103 - 1.61co103 = 0

45. 1,93Q13 + 1,93co13 + 1,42Q23 + 1.42co23 + 1.61Q33 + 1.61co33 + 1.87Q43 + 1,87co43 + 1,28Q53 + 1,28co53 + 1,61Q63 + 1,61co63 + 2.68Q73 + 2.68co73 + 4,88Q83 + 4,88co83 + 3,19Q93 + 3,19co93 + 1,61Q103 + 1,61co103 + H4 – uk4 – 1,92Q14 – 1,92co14 – 1,42Q24 – 1,42co24 – 1,61Q34 – 1,61co34 – 1,86Q44 – 1,86co44 – 1,28Q54 -1,28co54 – 1,61Q64 -1,61co64 – 2,65Q74 - 2,65co74 – 4,82Q84 – 4,82co84 – 3,16Q94 – 3,16co94 – 1,61Q104 – 1,61co104 = 0

46. 1,95Q11 + 1,95co11 + 1,43Q21 + 1,43co21 + 1,62Q31 + 1,62co31 + 1,88Q41 + 1,88co41 + 1.28Q51 + 1,28co51 + 1,62Q61 + 1,62co61 + 2,71Q71 + 2,71co71 + 4,94Q81 + 4,94co81 + 3,22Q91 + 3,22co91 + 1,62Q101 + 1,62co101 + ≤ 319700

47. 2,05Q12 + 2,05co12 + 1,47Q22 + 1,47co22 + 1,69Q32 + 1,69co32 + 1.98Q42 + 1,98co42 + 1.31Q52 + 1.31co52 + 1,69Q62 + 1.69co62 + 2,89Q72 + 2,89co72 + 5,36Q82 + 5,36co 82 + 3,46Q92 + 3,46co92 + 1,69Q102 + 1,69co102 ≤ 376400

48. 1,93Q13 + 1,93co13 + 1,42Q23 + 1,42co23 + 1,61Q33 + 1,61co33 + 1,87Q43 + 1.87co43 + 1,28Q53 + 1,28co53 + 1,61Q63 + 1,61co63 + 2,68Q73 + 2,68co73 + 4,88Q83 + 3,88co83 + 3,19Q93 + 3,19co93 + 1,61Q103 + 1.61co103 ≤ 314500

49. 1,92Q14 + 1,92co14 + 1,42Q24 + 1,42co24 + 1,61Q34 + 1,61co34 + 1,86Q44 + 1,86co44 + 1,28Q54 + 1,28co54 + 1,61Q64 + 1,61co64 + 2,65Q74 + 2,65co74 + 4,82Q84 + 4,82co84 + 3,16Q94 + 3,16co94 + 1,61Q104 + 1,61co104 ≤ 299000

Sonuçlar

Tablo 3.18. Doğrusal Optimizasyon Sonuçlar

Variable	**Value**	**Reduce Cost**	**Variable**	**Value**	**Reduce Cost**
Q11	3000.000	0.000000	Q81	2808.000	0.000000
Q21	2300.000	0.000000	Q91	1100.000	0.000000
Q31	5000.000	0.000000	Q101	490.000	0.000000
Q41	2300.000	0.000000	Q12	3200.000	0.000000
Q51	3600.000	0.000000	Q22	3000.000	0.000000
Q61	4420.000	0.000000	Q32	0.000000	0.012000
Q71	5700.000	0.000000	Q42	2700.000	0.000000
Q52	4800.000	0.000000	co71	0.000000	0.750000

Tablo 3.18. devam

Q62	5040.000	0.000000	co81	0.000000	0.970000
Q72	7200.000	0.000000	co91	0.000000	1.060000
Q82	2072.000	0.000000	co101	0.000000	1.040000
Q92	900.000	0.000000	co12	0.000000	0.650000
Q102	490.000	0.000000	co22	0.000000	1.000000
Q13	3390.000	0.000000	co32	0.000000	1.002113
Q23	1200.000	0.000000	co42	0.000000	1.160000
Q33	2400.000	0.000000	co52	0.000000	0.870000
Q43	2100.000	0.000000	co62	0.000000	9.930000
Q53	4500.000	0.000000	co72	0.000000	0.620000
Q63	4620.000	0.000000	co82	0.000000	0.790000
Q73	6400.000	0.000000	co92	0.000000	0.910000
Q83	2304.000	0.000000	co102	0.000000	1.210000
Q93	4900.000	0.000000	co13	0.000000	1.240000
Q103	630.000	0.000000	co23	0.000000	0.760000
Q14	610.000	0.000000	co33	0.000000	1.150000
Q24	3300.000	0.000000	co43	0.000000	1.710000
Q34	9900.000	0.000000	co53	0.000000	15.12000
Q44	10200.000	0.000000	co63	0.000000	0.740000
Q54	7500.000	0.000000	co73	0.000000	1.210000
Q64	7350.000	0.000000	co83	0.000000	1.120000
Q74	0.000000	9.562330	co93	0.000000	1.080000
Q84	8712.000	0.000000	co103	0.000000	0.900000
Q94	0.000000	9.537141	co14	0.000000	0.590000
Q104	2170.000	0.000000	co24	0.000000	1.220000
co11	0.000000	0.780000	co34	0.000000	1.080000
co21	0.000000	0.940000	co44	0.000000	15.618824
co31	0.000000	0.590000	co54	0.000000	10.223531
co41	0.000000	1.230000	co64	0.000000	1.050007
co51	0.000000	1.040000	co74	0.000000	10.653214
co61	0.000000	16.20000	co84	0.000000	1.140000

Tablo 3.18. devam

co94	0.000000	10.817513	S102	0.000000	5.996021
co104	0.000000	1.190000	S13	1790.000	0.000000
S11	0.000000	0.000005	S23	0.000000	0.000001
S21	0.000000	0.000004	S33	0.000000	0.000002
S31	2.400000	0.000005	S43	0.000000	0.000006
S41	0.000000	0.000008	S53	0.000000	0.000003
S51	0.000000	0.000001	S63	0.000000	0.000134
S61	0.000000	0.000117	S73	1700.000	0.000000
S71	0.000000	0.000000	S83	0.000000	0.000064
S81	376.0000	0.000000	S93	4100.000	0.000000
S91	0.000000	1.241000	S103	0.000000	1.280471
S101	0.000000	1.826000	S14	0.000000	12.10734
S12	0.000000	5.472365	S24	0.000000	10.10812
S22	0.000000	5.371280	S34	0.000000	9.436021
S32	0.000000	7.204102	S44	0.000000	13.109017
S42	0.000000	7.352174	S54	0.000000	10.228212
S52	0.000000	3.925290	S64	0.000000	76.382245
S62	0.000000	19.032061	S74	0.000000	5.307025
S72	0.000000	8.378696	S84	0.000000	24.607102
S82	0.000000	13.821053	S94	0.000000	8.935027
S92	0.000000	10.308572	S104	0.000000	11,963041
H1	69235.443	0.000000			
uk1	0.000000	4.881261			
H2	0.000000	0.782043			
uk2	0.000000	4.378552			
H3	8.406245	0.000000			
uk3	0.000000	3.372000			
H4	0.000000	0.000000			
uk4	0.000000	4.632410			

Problem çözüldüğünde amaç fonksiyonunun minimum maliyeti 1.180.086 TL olarak bulunur.

C) Bulanık ve doğrusal optimizasyonun karşılaştırılması

Tablo 3.19. Bulanık ve Doğrusal Optimizasyonun Karşılaştırılması

	Doğrusal Optimizasyon	Bulanık Optimizasyon	Kabul Aralığı
Maliyet (TL)	1.180.086	(1.167.184 - 1.178.900)	1.100.000-1.250.000

Dönemler için 1175000 TL lik maliyet hedefi ile bütçenin en fazla 750000 TL lik artması kabul edilebilir olarak belirlenmiştir. Bu 4 dönem boyunca taleplerde bir dalgalanma olması, bir talep artışının meydana gelmesi veya mevcut üretim süreçlerinde oluşabilecek bir problem (makine arızaları gibi) gibi durumların gerçekleşme olasılığı her zaman vardır. Bu nedenle belirlenen maliyet hedefine yaklaşma açısından, modelin derece başarılı olduğunu ya da hedefe ne derece yaklaşıldığını belirlenen toleransların önemi büyüktür. Çözümü gerçekleştirilen problemde ürünlere ait dönemlik bütçe için 750000 TL lik bir sapma payı öngörülmüş, fakat kurulan modelde belirtilen şartlarla (talep, makine, işçilik v.b.) λ = 0,156216 ile ayrılan 750000 YTL nin tamamının kullanılamayacağı sonucuna varılmıştır.

Önerilen çözüm metodu ile ulaşılan hedefin belirsiz olduğu unutulmamalıdır. Bilindiği gibi güncel hayatta ilgili veri ve model parametreleri farklılık gösterebilecektir. Önerilen yaklaşımda, maliyet minimizasyon hedefini sağlayabilmek için karar vericiye bir tatmin derecesi belirler. Bu dereceyi λ değeri gerçekleştirir, Yani elde edilen sonuçlar mevcut değişkenler ve bulanıklaştırdığımız kısıtların sağ taraf değişkenleri ile sağlanmıştır. Kıt kaynakların arttırılması ile amaç daha iyileştirebilir, λ düzeyi arttırılabilir. Bu nedenle amacı karşılayan çözüm elde edilinceye kadar, halihazırda olan belirsiz verilerle, ilgili model parametreleri sürekli değiştirilmelidir. Bulanık optimizasyon karar vericiye böyle bir imkan sunar. Doğrusal optimizasyonda ise güncel hayatta yaşanan belirsizliğin etkileri görülemez.

Deterministik olarak ifade edilen problemler için en iyi çözümü aranır, fakat her zaman gerçek hayatı yansıtma konusunda yeterli olunmaz. Bu nedenle yukarıda ilgilenilen problem kapsamında çözümü gerçekleştirmekte yetersiz kalınabilir.

Optimal çözüm değerleri, piyasa şartları, talep miktarları, üretim süreçlerinde ki kısıtların belirsiz olması nedeni ile güncel hayatta her zaman belirsizlik gösterecektir.

Bulanık optmizasyonun avantajları

- Günlük hayatta olduğu gibi belirsiz, zamanla değişen, karmaşık, iyi tanımlanmamış sistemlerin denetimine basit çözümler getirir.

- Bulanık mantık denetimi geleneksel mantığa göre sistemi daha iyi analiz edebileceği gibi aynı zamanda da ekonomiktir.

- Oldukça geniş bir alana yayılan değerlerin az sayıda üyelik fonksiyonlarına indirgenmeleri nedeni ile bulanık denetim genellikle daha küçük bir yazılımla daha hızlı bir şekilde sonuçlanır.

- Bu az sayıda değerler ile üzerinde uygulanacak kural sayısı da az olduğundan sonuca ulaşmak daha da çabuk olur. Bu durum geleneksel bilgisayar ortamında böyledir. Özel geliştirilmiş bir donanımla sonuca daha da hızlı ulaşmak olasıdır. Örneğin Sanyo-Fisher firması mühendisleri, video kayıt cihazında kullanmayı düşündükleri mikro bilgisayarın yetersiz kalmasından dolayı, bulanık denetim kullanmaya karar vermişlerdir. Bulanık denetim yazılım boyutlarının daha küçük olmasını sağladığından, dış bellek kullanımına gerek kalmamıştır.

Bilindiği gibi otomatik vites değişimi motorun belli hızlara ulaşması sonucunda otomatik olarak gerçekleşir. Buna karşılık manüel vitesli bir arabada ise sürücü, yol, yük ve kendi araba kullanış tarzına göre belli durumlarda vites değiştirir. Subaru tarafından üretilen Justy tipi otomobilde kullanılan aktarım organının değiştirilmesi, bir kayışın konumunun bulanık mantık kullanılarak değiştirilmesi ile sağlanır. Böylece arabanın ivmesi ve performansı sürekli olarak ayarlanır hale gelir. Subaru, bu otomobilde kullandığı bulanık mantık üyelik fonksiyonlarını, otomobili test şoförlerine kullandırarak ve onlardan ivme ve performans açısından en iyi aktarım oranını öğrenerek ayarlamıştır. Bu konuda Honda, Nissan ve Toyota da benzer çalımsalar yapılmıştır.

Bulanık optmizasyonun dezavantajları

- Uygulamada kullanılan kuralların oluşturulmasının bulanıklılığa bağlılığı,
- Üyelik fonksiyonlarının deneme – yanılma yolu ile bulunmasından dolayı uzun zaman alabilmesi,
- Kararlılık analizinin yapılışının zorluğu, Denetlenen sistemin bir kararlılık analizi yapılamaz ve sistemin nasıl cevap vereceği önceden kestirilemez. Yapılacak tek şey benzetim çalışmasıdır.

İyi bir mühendislik teorisinin incelenen olayın önemli bazı özelliklerini yakalayarak onu yaklaşık bir biçimde modellemesi ve matematik bakımından karmaşık olmayacak çözümlerle kontrol altına alınması beklenir. Aslında bulanık yöntemlerle bir sistemin modellenmesinde de yaklaşıklık ve oldukça kolay çözünürlük bulunur. Mühendislik yaklaşımlarında elde edilen tüm sayısal ve sözel bilgiler çözüm algoritmasına katılarak incelenen olayın kontrolünde anlamlı çözümlere varılabilmelidir. Bu bakımdan bulanık küme, mantık ve sistem ilkeleri, uzman kişilerinde vereceği sözel bilgileri işleyerek toptan çözüme gitmeye yarar. Halbuki, teorik matematik ve diferansiyel hesaplamalarda sadece sayısal değerler kullanılır. İnsanların sunduğu sözel bilgilerin sayısal hale getirilerek bilgisayarlar veya algoritmalar tarafından algılanarak hesaplamaların yapılabilmesi için bulanık sistemlere gerek vardır.

3.5. Sürü Optimizasyonu (Swarm Optımzatıon) Algoritması

Hesaplama alanında kullanılan ve sürülerden esinlenilerek ortaya konulan iki popular metod vardır. Bunlar " karınca koloni optimizasyonu" (KKO) ve " sürü optimizasyonu" (SO) dur. Sürü Optimizasyonu (swarm optimization) sürü davranışlarından esinlenilerek ortaya çıkarılmış en yeni optimizasyon tekniklerinden birisidir. 1995 yılında J.Kennedy ve R.C.Eberhart tarafından; kus sürülerinin davranışlarından esinlenilerek ortaya çıkarılmış populasyon tabanlı optimizasyon tekniğidir [54].

Algoritmayı uygulamak oldukça basittir ve çok parametreli, doğrusal olmayan problemlerin çözümü için kullanılmaktadır.

3.5.1. Sürü zekası

Biyolojik sistemlerden esinlenilerek ortaya çıkarılmış birçok yöntem, hesaplama problemlerinin çözümünde kullanılmaktadır. Örneğin yapay sinir ağları insan beyninin basitleştirilmiş bir modelidir, genetik algoritma ise insan evriminden esinlenilerek ortaya çıkarılmıştır. Biyolojik sistemlerin başka bir çeşidi olan sosyal sistemler ise özellikle bireyin çevresiyle ve diğer bireylerle olan etkileşimi ve kolektif (ortak) davranışları incelemektedir. Bu davranışlar sürü zekası olarak adlandırılmaktadırlar. Sürü zekası (Swarm Intelligence); yetenekleri kısıtlı canlıların bir takım haline geldiklerinde üstün yetenek ve yönetim gerektiren davranışlar sergilemesidir. Karıncaların yiyeceğe giden en kısa yolu bulmaları, kuşların sürü halindeyken ki uçuş şekilleri ve yine karıncaların aralarında görev paylaşımı yapmaları ve kendilerini örgütleyebilmeleri sürü zekasına örnek olarak gösterilebilir.

Sürü zekasının en önemli özelliklerinden birisi merkezi bir yönetimin olmamasıdır (decentralization). Sürüde yer alan bireylerin çok basit karar mekanizmaları vardır ve sadece o işi yapmakla kendilerini yükümlü kılarlar. Her etmenin kendi işini yapması sonucu ortaya ancak karmaşık bir karar mekanizmasına sahip bir yöneticinin vereceği kararla gerçekleşecek eylemler çıkar. Sürü zekasının diğer bir özelliği de kendini örgütleyebilen (self organisation) bireylerden oluşmasıdır. Sürü bireyleri,

değişen şartlara göre kendilerini örgütleyebilirler, yaptıkları işin cinsini değiştirme ve duruma göre kendilerine uygun işi tayin edebilme yeteneklerine sahiptirler. Aşağıda sürü zekasına ait bazı özellikler verilmiştir.

Kendi aralarında lokal iletişime sahip çok sayıda bireyden oluşur.
Otonom bir sistemdir tüm bireyler kendi kontrolünü yaparlar.
Ortak bir amaç doğrultusunda bir arada bulunurlar.
Tüm bireyler birbirlerinin davranışlarından etkilenirler.
Merkezi bir sistem tarafından yönetilmezler.
Bireylerin çevre ile iletişimleri vardır. Ortam bilgilerini elde edebilirler.
Birimlerden bazıları çalışmaz hale gelse bile sistem devam ettirilebilir.

3.5.2. Sürü algoritması

Sürüü Optimizasyonu (SO); 1995 yılında J.Kennedy ve R.C.Eberhart tarafından; kuş sürülerinin davranışlarından esinlenilerek geliştirilmiş populasyon tabanlı rassal arama algoritmasıdır. Doğrusal olmayan problemlerin çözümü için tasarlanmıştır. Çok parametreli ve çok değişkenli optimizasyon problemlerine çözüm bulmak için kullanılmaktadır. SO, genetik algoritmalar gibi evrimsel hesaplama teknikleriyle birçok benzerlik gösterir. Sistem rasgele çözümler içeren bir populasyonla başlatılır ve nesilleri güncelleyerek en optimum çözümü araştırır. SO da parçacık olarak adlandırılan olası muhtemel çözümler, o andaki optimum parçacığı izleyerek problem uzayında dolaşırlar.

SO'nun klasik optimizasyon tekniklerinden en önemli farklılığı türev bilgisine ihtiyaç duymamasıdır. Bu özellik birçok problemin çözümü için gerekli olan karmaşık işlem yükünün hafifletilmesini sağlamaktadır. SO'yu uygulamak, ayarlanması gereken parametre sayısının az olması sebebiyle oldukça basittir. SO; fonksiyon optimizasyonu, bulanık sistem kontrolü, yapay sinir ağı eğitimi gibi birçok alanda başarıyla uygulanabilmektedir [55-56].

Yukarıda da belirtildiği gibi SO kuş sürülerinin davranışlarının bir benzetimidir. Kuşların uzayda, yerini bilmedikleri yiyeceği aramaları, yiyecek ararken yiyeceğe en

yakın olan kuşu takip etmeleri gibi. SO'da popülasyonu oluşturan bireyler parçacık olarak adlandırılır, her bir parçanın durum uzayında hareket ettiği varsayılır ve her parça potansiyel çözümü taşır. Her parça en iyi durumu hatırlayabilir ve parçacıklar kendi arasında bilgi alışverişinde bulunabilirler [54]. Parçacık olarak adlandırılan her tekil çözüm, arama uzayındaki bir kuştur. Parçacık hareket ettiğinde, kendi koordinatlarını bir fonksiyona gönderir ve böylece parçacığın uygunluk değeri ölçülmüş olur. (yiyeceğe olan uzaklık değerinin belirlenmesi) Bir parçacık, koordinatlarını, hızını (çözüm uzayındaki her boyutta ne kadar hızla ilerlediği), şimdiye kadar elde ettiği en iyi uygunluk değerini ve bu değeri elde ettiği koordinatları hatırlamalıdır. Çözüm uzayında her boyuttaki hızının ve yönünün, her seferinde nasıl değişeceği, komşularının en iyi koordinatları ve kendi kişisel en iyi koordinatlarının birleşimi olacaktır. Problemin çözüm uzayı, değişken veya bilinmeyen sayısına bağlı olarak çok boyutta olabilir [57-58].

Literatürde SO birçok problemin optimizasyonunda başarı ile kullanılmıştır,

1. Fonksiyon optimizasyonu
2. Optimal sipariş miktarı belirleme,
3. Sinir ağları,
4. Akış tipi çizelgeleme problemleri, güç ve voltaj kontrolü,
5. Tedarik seçimi ve sıralama problemi [54]

Örnek verilecek olursa f $(X_1, X_2, X_3, X_4) = 3{X_1}^2 + 4X_2 + X_3 - {X_4}^4$ fonksiyonunun çözüm uzayı bilinmeyenlerinden dolayı 4 boyutludur. Bu problemin çözüm uzayında tanımlanan bir parçacığın pozisyonu 4 koordinat ile P = $[X_1, X_2, X_3, X_4]$ şeklinde belirtilmektedir. Örneğin bu fonksiyonda çözümlerden biri P = [4 ,7, 5, 3] dir fakat problemi ifade eden fonksiyon veya uygunluk değeri, SO algoritması veya parçacıklar için önem arz etmez.

SO da bir grup rasgele çözümle (parçacık sürüsü) başlatılır ve güncellemelerle optimum çözüm bulunmaya çalışılır. Her iterasyonda, parçacık konumları, iki en iyi değere göre güncellenir. İlki o ana kadar parçacığın elde ettiği en iyi çözümü sağlayan koordinatlardır. Bu değer "pbest" olarak adlandırılır ve hafızada

saklanmalıdır. Diğer en iyi değer ise, populasyonda o ana kadar tüm parçacıklar tarafından elde edilen en iyi çözümü sağlayan koordinatlardır. Bu değer global en iyidir ve “gbest” ile gösterilir. N adet parametreden oluşan n adet parçacık olduğunda, populasyon matrisi aşağıdaki gibi olur.

$$S = \begin{bmatrix} S_{11}, S_{12}, S_{13}, & \cdots & S_{1N} \\ \vdots & \ddots & \vdots \\ S_{n1}, S_{n2}, S_{n3} & \cdots & S_{nN} \end{bmatrix}$$

Matriste i. parçacık $S_i = [S_{11}, S_{12}, S_{13}, \dots S_{1N}]$

Önceki en iyi uygunluk değerini veren i’ninci parçacığın pozisyonu:

$\text{pbest}_i = [p_{i1}, p_{12}, p_{13}, \dots p_{1N}]$

i’ninci parçacıgın hızı $V_i = [V_{11}, V_{12}, V_{13}, \dots V_{1N}]$

gbest tüm iterasyonlarda her parçacık içi tektir.

3.5.3. Sürü optimizasyonunun parametreleri

SO da ayarlanması gereken çok fazla sayıda parametre yoktur ancak parametrelerin seçimi oldukça önemlidir. Çözülecek probleme özgü, uygun parametreler seçildiği takdirde algoritmanın başarısı artmaktadır [59-60].

Parçacık Sayısı: 20 ile 40 arasındadır. Birçok problem için 10 parçacık kullanmak iyi sonuç elde etmek için yeterlidir. Bazı özel problemlerde ise 100 parçacık kullanılması gerekebilir.

Parçacık Boyutu: Optimize edilecek probleme göre değişmektedir. (çözüm uzayının boyutu)

Parçacık Aralığı (Range): Optimize edilecek probleme göre değişmekle birlikte farklı boyutlarda ve aralıklarda parçacıklar tanımlanabilir.

Vmax: Bir iterasyonda, bir parçacık da meydana gelecek maksimum değişikliği (hız) belirler. Genellikle parçacık aralığına göre belirlenir. Örneğin X_1 parçacığı (-10,10) aralığında değişiyor ise Vmax = 20 sınırlandırılabilir.

Öğrenme Faktörleri: c1 ve c2 genellikle 2 olarak alınır, fakat farklı da seçilebilir. Genellikle c1, c2 ye eşit ve (0-4) aralığında seçim yapılır.

Eylemsizlik Ağırlığı: w; her iterasyonda doğrusal olarak azaltılmalıdır. Genellikle $0.8 < w < 1.2$ aralığında alınır.
Durma Koşulu: İstenilen uygunluk değerine veya maksimum iterasyon sayısına ulaşıldığında durdurulabilir. Durdurma koşulu optimize edilecek probleme bağlıdır

3.5.4. Sürü optimizasyon algoritması

SO' da her bir parçacığın kendine ait bir hızı vardır ve bu hız parçacığı diğer parçalardan aldığı bilgilerle optimum sonuca doğru hızlandırır. Her bir jenerasyonda ki hız önceki en iyi sonuçlardan da faydalanarak yeniden hesaplanır. Algoritma adımları şu şekildedir;

1. Popülasyonun oluşturulması; parçacıklar, rastgele üretilen başlangıç pozisyonları ve hızları ile birlikte oluşturulur.

2. Uygunluk değerlerinin hesaplanması; popülasyon içindeki tüm bireylerin uygunluk değerleri hesaplanır.

3. En iyi üyenin bulunması; her jenerasyonda bütün bireyler bir önceki jenerasyondada bulunan en iyi (pbest) ile karşılaştırılır. Eğer daha iyi birey varsa yer değiştirilir.

4. Global en iyinin bulunması; jenerasyondaki en iyi değer global en iyi değerden daha iyi ise yer değiştirilir.

5. Pozisyon ve hızların yenilenmesi;

$$V_{iN} = W.V_{iN} + c_1.rnd_1.(p_{iN} - X_{iN}) + c_2.rnd_2.(p_{iN} - X_{iN})$$

X_{iN}: Pozisyon Değeri

V_{iN}: Hız Değeri

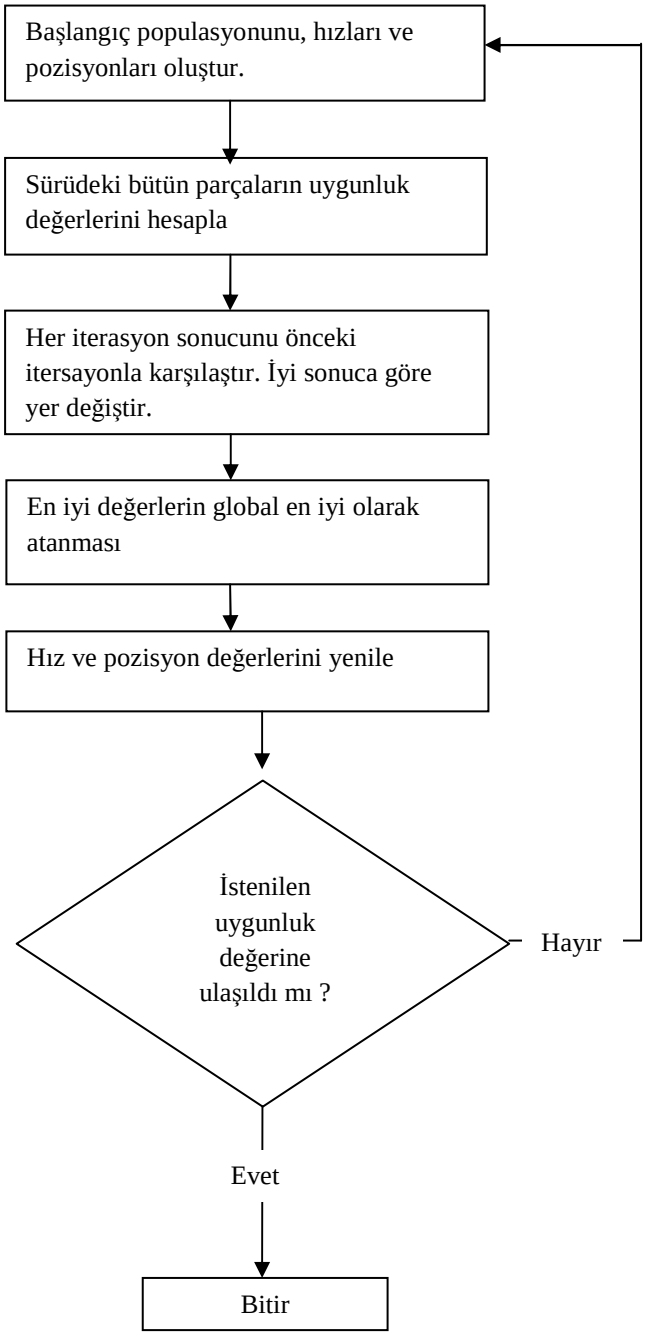
rnd_1, rnd_2: 0 – 1 arasında üretilmiş rassal say

W: Eylemsizlik Ağırlığı ($0.8 < w < 1.2$)

c_1, c_2: Sabit değerler genellikle 2 alınır.

6. Durdurma kriteri sağlanıncaya kadar adımları tekrar et. Durdurma kriteri sağlanıncaya kadar adım 2-5 tekrar et.

SO algoritmasının akış diyagramı aşağıda şekil 3.38 de ki gibidir.



Şekil 3.39. SO Algoritmasının Akışı

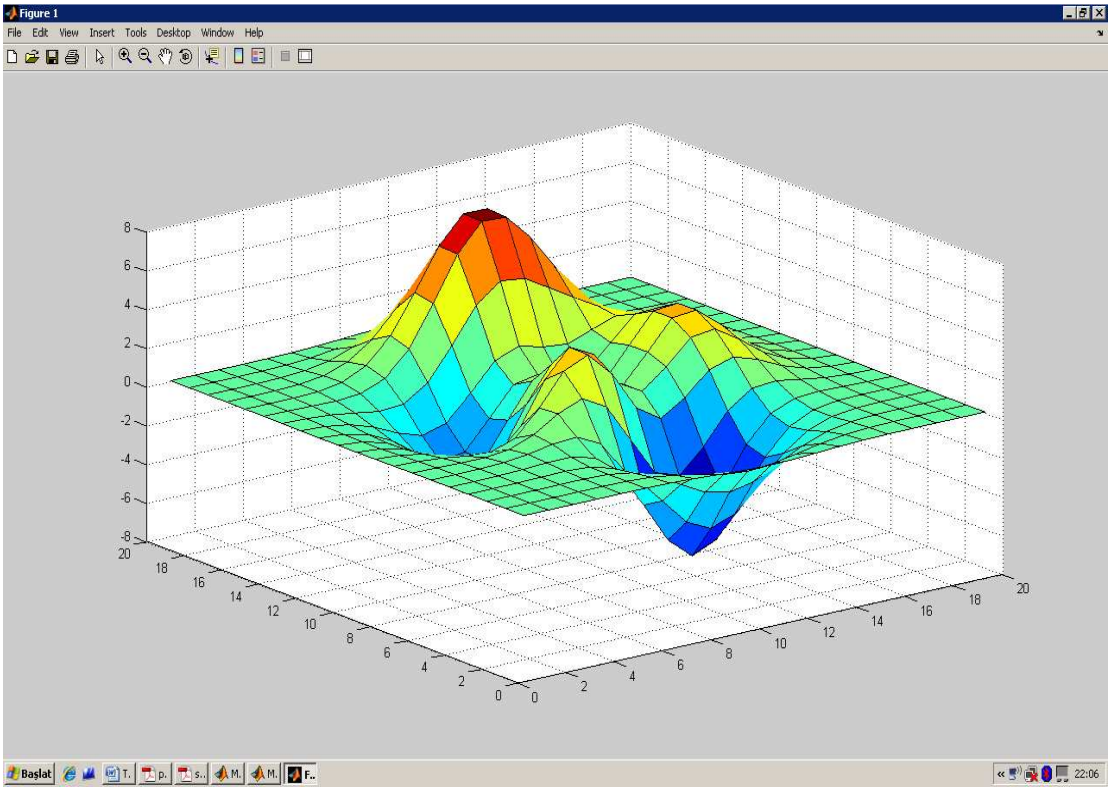
3.5.5. Sürü optimizasyonu ile matlabda fonksiyon minimizasyonu

$$f(x_1x_2) = 3.(1-x_1)^2.e^{(-{x_1}^2-(x_2+1)^2)} - 10.\left(\frac{x_1}{5} - {x_1}^3 - {x_2}^5\right).e^{(-{x_1}^2-{x_2}^2)} - \frac{1}{3}.e^{(-{x_2}^2-(x_1+1)^2)}$$

Bu fonksiyon x (x_1) ve y (x_2) değişkenleriyle Matlab ortamında aşağıdaki gibi ifade edilir.

```
3*(1-x).^2.*exp(-(x.^2) - (y+1).^2) - 10*(x/5 - x.^3 - y.^5).*exp(-x.^2-y.^2) - 1/3*exp(-(x+1).^2 - y.^2)
```

Matris olarak z gibi bir matris değişkenine **z=peaks(n);** komutuyla atayıp z matrisi ile ilgili grafikler çizdirilebilir. Örneğin; **surface(z)** ile z nin iki boyutlu alan grafiği; **surf(z)** ile z nin yüzey grafiği, **mesh(z)** ile ağ grafiği, **contour(z)** ile yüzey seviye grafiği çizdirilebilir.



Şekil 3.40. $f(x_1x_2)$ Fonksiyonunun Grafiği

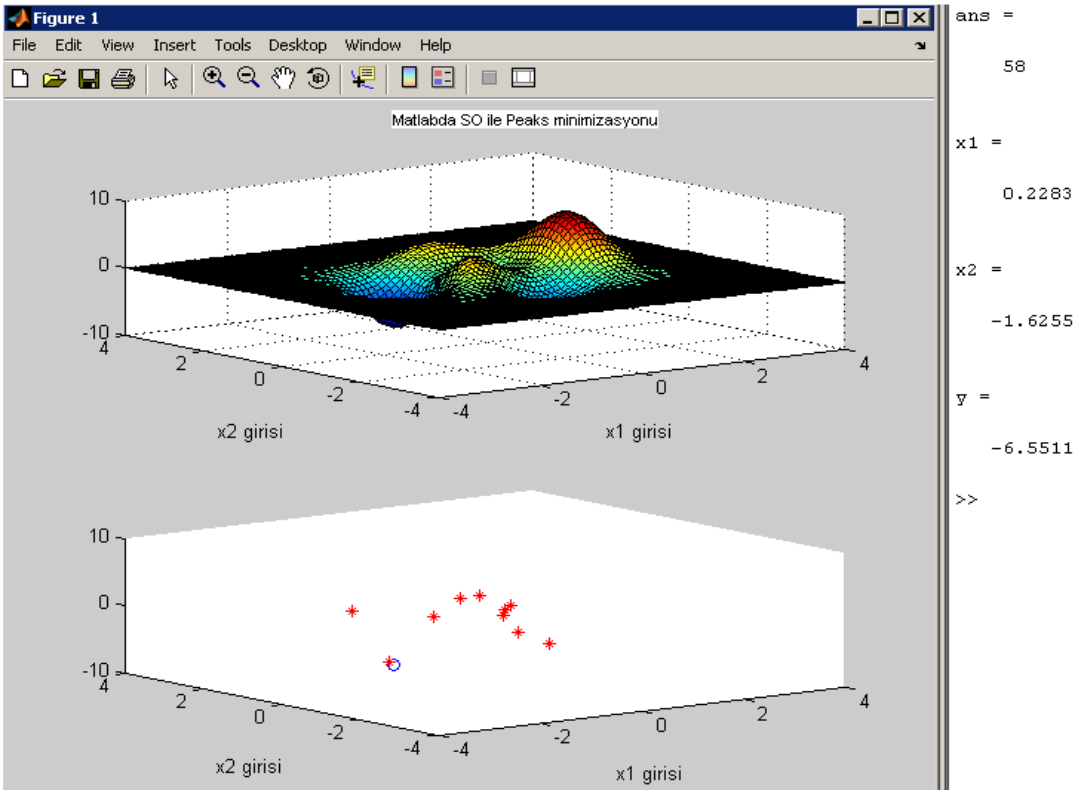
```
>> clear all; close all; clc;
x1=-4:0.1:4;
x2=-4:0.1:4;
for i=1:1:81
for j=1:1:81
y(i,j)=3*(1-x1(i))^2*exp(-x1(i)^2-(x2(j)+1)^2)...
-10*((1/5)*x1(i)-x1(i)^3-x2(j)^5)*exp(-x1(i)^2-x2(j)^2)...
-(1/3)*exp(-(x1(i)+1)^2-x2(j)^2);
end
end
subplot(2,1,1); surf(x1,x2,y)
xlabel('x1girisi'); ylabel('x2 girisi'); zlabel('h')
p=10; %% parçacık sayısı
P=randn(p,2); %% random atama
% P=[-1.5 0.5; %% local minimumluk
% -1.5 -0.5;
% -1.5 0;
% -1.3 0.5;
% -1.3 -0.5;
% -1.3 0;
% -1.7 0.5;
% -1.7 -0.5;
% -1.7 0;
% -1.4 0.5];
P0=P; pbest=P;
c1=2; c2=2; %% Öğrenme etkenleri
%% Her parçada başlangıç y nin hesaplanması
%% hesaplanan en küçük y değerleri gecmisy
for n=1:1:p
y(n)=3*(1-pbest(n,1))^2*exp(-pbest(n,1)^2-(pbest(n,2)+1)^2)...
-10*((1/5)*pbest(n,1)-pbest(n,1)^3-pbest(n,2)^5)...
*exp(-pbest(n,1)^2-pbest(n,2)^2)-...
```

```
(1/3)*exp(-(pbest(n,1)+1)^2-pbest(n,2)^2);
gecmisy(n)=y(n);
end
subplot(2,1,2); plot3(P(:,2),P(:,1),gecmisy,'r*');hold on;
Vm=randn(p,2); %% parametrelerdeki değişim
gbest=randn(1,2); %% pbestlerin en iyisi
w=0.9;
for k=1:1:100 %% iterasyon sayacı
w=w-0.009;
for n=1:1:p %% p=10 tane parca
y(n)=3*(1-P(n,1))^2*exp(-P(n,1)^2-(P(n,2)+1)^2)...
-10*((1/5)*P(n,1)-P(n,1)^3-P(n,2)^5)*exp(-P(n,1)^2-P(n,2)^2)...
-(1/3)*exp(-(P(n,1)+1)^2-P(n,2)^2);
if (y(n)<gecmisy(n))
gecmisy(n)=y(n);
pbest(n,:)=P(n,:);
end
end
for n=1:1:p
[a b]=min(gecmisy);
gbest(k,:)=pbest(b,:);
pb=sum(pbest)/p;
Vm(n,:)=w*Vm(n,:)+c1*rand*(pb-P(n,:))+c2*rand*(gbest(k,:)-P(n,:));
P(n,:)=P(n,:)+Vm(n,:);
end
end
x1=gbest(k,1)
x2=gbest(k,2)
y=3*(1-x1)^2*exp(-x1^2-(x2+1)^2)-10*((1/5)*x1-x1^3-x2^5)*exp(-x1^2-x2^2)...
-(1/3)*exp(-(x1+1)^2-x2^2)
subplot(2,1,2);
plot3(P(:,2),P(:,1), gecmisy,'bo'); axis([-4 4 -4 4 -10 10]);
```

xlabel('x2 girisi')

ylabel('x1 girisi')

Yazılan programın elde edilen sonucu aşağıda şekil 3.40 da ki gibidir.



Şekil 3.41. f($x_1 x_2$) Fonksiyon Sonuç

$$\text{Min f } (x_1 \ x_2) = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 0{,}2283 \\ -1{,}6255 \end{bmatrix} = -6{,}5511$$

$$\text{Max f } (x_1 \ x_2) = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} -0{,}0092 \\ 1{,}5815 \end{bmatrix} = 8{,}1052 \text{ olarak bulunur}$$

Yukarıdaki ekran çıktısında (şekilde) fonksiyon yüzeyinin ve parçacık pozisyonlarının x düzleminden görünüşü verilmiştir. Kırmızı renkli yıldız işareti ile gösterilmiş olanlar parçacıkların başlangıç noktalarını göstermekte, mavi renkli yuvarlak işareti parçacıkların minimum çözümünde son adımda geldikleri noktayı şekil 3.40 da görüleceği üzere parçacıkların hepsi sürü mantığına uygun olarak global çözümün etrafında toplanmışlardır.

SO algoritması klasik yöntemlere alternatif olarak ortaya konmuş bir yöntemdir. Algoritma; herhangi bir diferansiyel denklem çözümü ve gradyan hesabı gerektirmediğinden çok boyutlu problemlere oldukça hızlı ve yeterli sonuçlar verdiği görülmektedir. Bu nedenle gradyan tabanlı klasik öğrenme algoritmaları yerine SO' nun kullanılabileceği söylenebilir.

3.6. Tavlama Benzetimi Algoritması (Simulated Annealing)

Bu algoritma GA da ki evrimsel benzetimden esinlendiği gibi, katıların belirli bir başlangıç sıcağından başlayarak yavaş yavaş soğutulduğu tavlanma sürecinin benzetimi olan stokastik bir arama algoritmasıdır. Vecchi, Gerlatt ve Kirkpatrick tarafından ortaya konulmuş, ardından da Cerny tarafından kullanılması önerilmiştir. Son yılarda birçok araştırmacı TBA kombinatoriyel problemlerde en iyileme aracı olarak kullanmaktadır.

Maddenin sıvı durumunda tüm atomlar gelişi güzel hareket ederler. Katı durumda ise atomların hareket alanı oldukça kısıtlıdır. Bu durumda sistemin enerjisi en azdır. Tavlama terimi, ısıtılmış yüksek enerjili bir cismin yavaş yavaş soğumaya bırakılarak düşük enerjili konuma geçmesi ve kristal yapıya geçene kadar soğutulmasıdır. Eğer cisim hızlı bir şekilde soğutulursa, yapısında ilerde kırılmalara yol açabilecek istenmeyen çatlaklar oluşacaktır.

TBA' nın diğer optimizasyon problemleri ile benzer yönleri vardır. Bu benzerlikler aşağıdaki gibidir:

Maddenin farklı fiziksel durumları ⟶ Problemdeki mümkün çözümlere
Sistemin enerjisi⟶ Amaç fonksiyonuna
Tam kararlı olmayan durum ⟶Yerel en iyi çözüme
Kararlı durum (maddenin enerjisinin en az olduğu durum) ⟶ Genel en iyi çözüme benzetilir.

Tablo 3.7. Tavlama benzetimi ve klasik optimizasyon

TAVLAMA BENZETİMİ	KLASİK OPTİMİZASYON
Sistem Durumları	Uygun Çözümler
Enerji	Amaç fonksiyonu
Durumun Değişimi	Yerel Çözüm
Sıcaklık	Kontrol parametresi

Maddenin katı halinin tavlanma sürecinde geçirdiği evreleri benzetebilmek amacıyla Metropolis Algoritması geliştirilmiştir. Bu algoritmada i durumunda bulunan katının enerjisi E_i, bir sonraki j durumuna geçerken olan enerjisi E_j olarak gösterilir. Eğer j durumundaki enerji, i durumundaki enerjiden küçük veya eşitse j durumu mevcut çözüm olarak kabul edilir. Eğer bunun tersi ise j durumu aşağıda gösterilecek olan olasılık değeri ile kabul edilir. Bu olasılık değerine Metropolis kriteri denir.

$$p(E_j) = e^{-\left[\frac{E_i - E_j}{k_B.T}\right]}$$

$p(E_j)$: j durumları içinden yukarıdaki fonksiyona göre optimal çözüm çıkma olasılığı

E_i : Bir önceki ana kadar olan çözüm değeri

E_j : Elde edilen çözüm değeri

T : O an ki sıcaklık değeri

k_B: Boltzmmann sabiti

TBA rassal olarak seçilen bir başlangıç çözümüyle aramaya başlar. Fonksiyonda meydana gelen değişim hesaplanır. Eğer değişim istenilen yönde ise bu çözüm durumu mevcut çözüm olarak alınır. İstenilen yönde bir değişim sağlanamamışsa yukarıda ki Metropolis olasılık sürecinden geçirilir ve bu süreçte elde edilen bir olasılık değeri ile kabul veya red gerçekleşir. Belirtilen süreçte olasılık değerine göre T değeri yüksek olduğunda, kabul edilme oranı artar, T değeri azaldıkça kabul edilme oranı da azalacaktır.

Bundan dolayı algoritmada yerel noktalara takılmamak için başlangıç sıcaklığı yüksek seçilerek sıcaklık yavaş yavaş azaltılır. Yukarıda belirtildiği üzere istenmeyen çatlakların önüne geçmek için soğuma yavaş yapılır. Algoritmanın genel yapısı aşağıdaki gibidir.

Bir başlangıç durumu seç;

Bir başlangıç sıcaklığı seç T > 0;

Sıcaklık değişim sayısını sıfırla t = 0;

Tekrarla

Tekrar sayacını sıfırla n = 0;

Tekrarla

i'nin bir komşusu olan j durumunu üret;

Δ = E(j) - E(i) ;

if Δ< 0 then i = j;

Else

rastsal sayı üret (0,1) <$e^{-\left[\frac{E_i - E_j}{k_B \cdot T}\right]}$) then i:= j;

n = n+1;

n= N(t);

t = t+1;

T = T(t);

Durdurma koşulu

End

Her hangi bir problemin çözümünde TBA' nın kullanılması için bazı parametrelerin belirlenmesi gerekir. Bu parametreler:

Başlangıç sıcaklığı (T_0),
Her sıcaklıktaki iterasyon uzunluğu,
Soğutma fonksiyonu, (sıcaklık azaltım fonksiyonu)
Algoritmayı durdurma kriteri

Başlangıç sıcaklığı bir girdi parametresidir. Sıcaklık kötü çözümlerin kabul edilme olasılığını kontrol etmek için kullanılır. İterasyon uzunluğu, her sıcaklıkta üretilen çözümlerin sayısıdır. Soğutma fonksiyonu (sıcaklık azaltım fonksiyonu) bir önceki iterasyon sıcaklığına bağlı olarak mevcut iterasyondaki sıcaklığı belirler. Soğutma oranı *r* olarak gösterilir. Uygulamalarda *r* değeri genellikle 0,8 ve 0,99 arasında alınmaktadır. Başlangıç sıcaklığı ile birlikte iterasyon sayısı ve soğutma fonksiyonu, soğutma çizelgesi olarak adlandırılır. Bu çizelge, çözüm kalitesinde veya yakınsama oranında büyük etkiye sahiptir. Her sıcaklık değişiminde elde edilen çözüm, çok sayıda ardışık sıcaklık değişimlerinde değişmiyor ise TBA durdurulur.

3.7. Karınca Koloni Optimizasyonu (Ant Colony Optimization)

Karıncalar görme duyuları gelişmiş olmamasına rağmen, yuvaları ile besin kaynakları arasındaki en kısa yolu bulabilmektedirler. Bu davranış kalıplarının, özellikle en kısa yol problemleri olmak üzere, pek çok kombinatoriyel optimizasyon probleminde kullanılabileceği ilk kez 1992 yılında Marco Dorigo tarafından ortaya atılmıştır. Karınca Kolonileri Meta sezgiselinden türetilmiş ve çeşitli problemlerin çözümünde kullanılan çok sayıda algoritma vardır. Bu algoritmalar ile birçok kombinatoriyel optimizasyon problemleri çözülmüştür. Örneğin Yapay sinir ağları, Genetik Algoritmalar, Metropolis algoritması, sürü algoritması optimizasyonları bu şekilde bulunmuş yöntemlerdir. Karınca koloni optimizasyonu da yuvaları ile besin kaynakları arasında izledikleri yolların izlenmesi sonucu ortaya çıkan bilimsel gerçekler üzerine doğmuştur.

Gerçek karıncalar ile ilgili deneyler Goss ve arkadaşları tarafından 1989 yılında laboratuar ortamında yetiştirilmiş karınca kolonileri üzerinde yapılmıştır. Bu çalışmalarda elde edilen sonuçlar şöyledir [61].

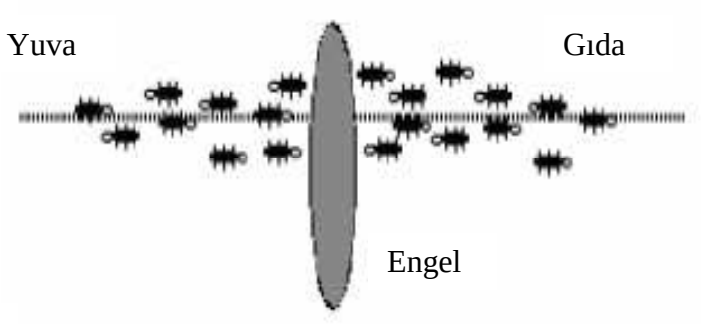
Pek çok karınca türü neredeyse kördür.

Karıncalar yuvalarından yiyecek kaynağına veya tersi yönde hareket ederlerken geçtikleri yollara feromen adı verilen bir tür kimyasal madde bırakmaktadırlar,

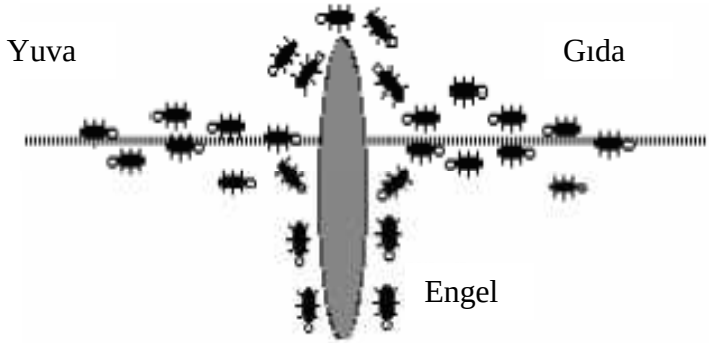
Karıncalar bir yol seçmeleri gerektiği zaman bu seçimi alternatif yollar üzerine bırakılmış olan feromen madde yoğunluğuna göre belirlemektedirler

Karıncaların bu hareketleri merkezi bir kontrol ile sağlanmamaktadır şeklinde özetlenmiştir.

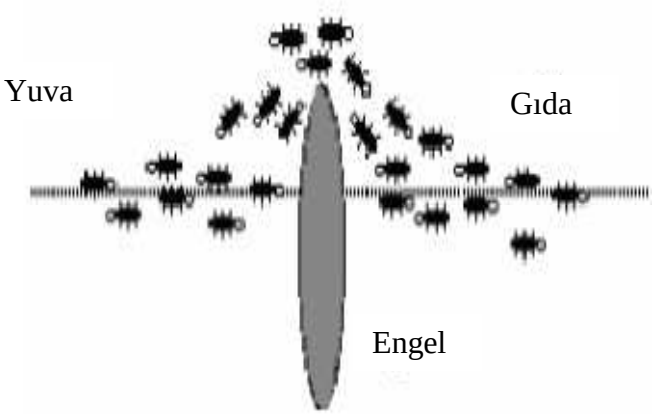
Karıncaların yuvaları ile yiyecek kaynağı arasındaki hareketleri aşağıdaki şekillerde gösterilmiştir [62].



Şekil 3.42. Karıncaların İzlediği Yol



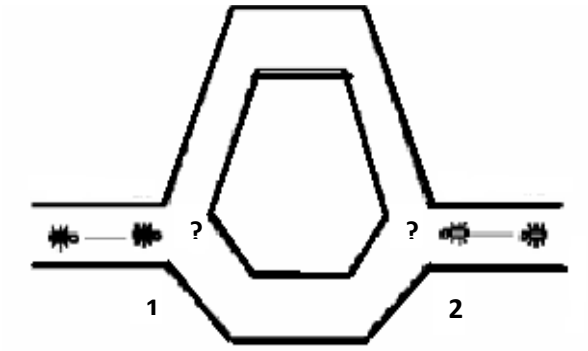
Şekil 3.43. Karıncaların Bir Engelle Karşılaşması



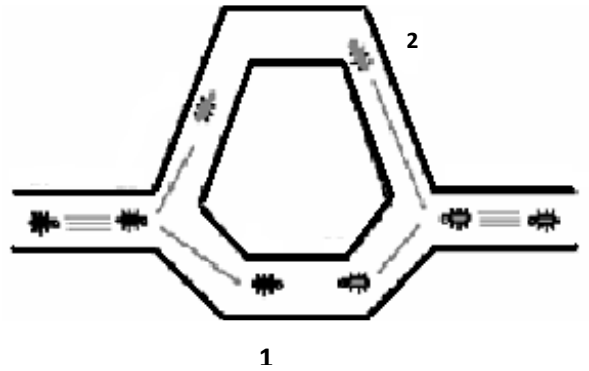
Şekil 3.44. Engelle Karşılaşan Karıncaların Seçimi

Şekillerde görüldüğü gibi, karıncalar yuvalarının etrafındaki alanda yiyecek kaynaklarını rassal bir şekilde ararlar. Bir karınca bir yiyecek kaynağı bulduğu zaman kaynağın kalitesini veya miktarını değerlendirir ve bir miktar yiyecek alarak

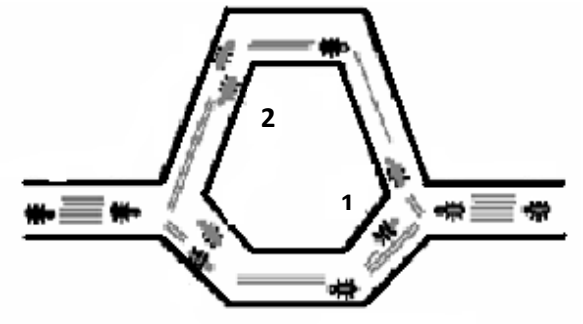
yuvasına geri döner. Bu geri dönüş sırasında, bulduğu yiyecek kaynağının kalitesi veya miktarıyla doğru orantılı olacak şekilde kullandığı yola **feromen** maddesi koyar. Böylece diğer karıncalar bu yolun sonundaki yiyecek kaynağının kalitesi veya miktarı konusunda bilgi sahibi olurlar. Yuvaya yakın kaynaklara ulaşmak daha kolay olacağı için bu bölgelerde feromen maddesinin yoğunluğu daha fazla olacaktır. Karıncaların bu hareketlerinin sayısal örneği aşağıdaki şekilde verilmiştir.



Şekil 3.45. Karıncaların Bir Karar Noktasına Gelmeleri

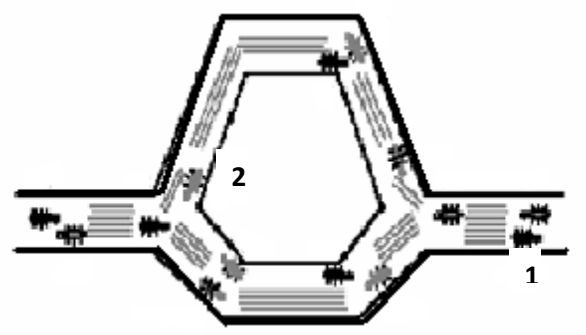


Şekil 3.46. Seçimin Tümüyle Rassal Olması Ve Farklı Yolların Seçilmesi



Şekil 3.47. Bir Sonra Ki Karar Noktasına Ulaşma

Birbirlerine yakın sabit bir hızda yürüdükleri için aşağıdaki ve kısa olan yolu seçen karıncalar turlarını daha çabuk bitirirler ve bir sonraki karar noktasına daha çabuk ulaşırlar.



Şekil 3.48. Kısa Yollarda Oluşan Feromen Miktarı

3.7.1. Sistemin formülasyonu

Karınca sistemi ile ele alınan bir problemin sonucunun, o problemi meydana getiren n adet varlığın (entity) permutasyonu olduğu varsayılır (gezgin satıcı problemindeki şehirler ya da atölye tipi çizelgeleme problemindeki operasyonlar gibi). Problemin başlangıcında her karınca farklı veya aynı köşelere yerleştirilir. Bu karıncalar (t) anında hangi komşu düğüm noktasında olacakları aşağıdaki olasılık bağıntısına göre belirlenir [62].

$$P_{ij}{}^{k}(t) = \begin{cases} \dfrac{\left(t_{ij}(t)\right)^{\alpha} . (n_{ij})^{\beta}}{\sum\left(t_{ik}(t)\right)^{\alpha} . (n_{ik})^{\beta}}, & k \text{ izinli bir seçimse} \\ 0, & \text{diğer durumlarda} \end{cases}$$

$t_{ij}(t)$: t anında (i, j) köşelerindeki feromen izi miktarı

n_{ij}: (i, j) köşeleri arasındaki görünürlük değeri

α: Problemde feromene gösterilen bağıl önem derecesi veya parametresi ($2 < \alpha < 4$)
α değeri, ilgili yolun feromon miktarının önemini belirler ve önceki iterasyonların sonuçlarının ilerleyen iterasyonlara aktarılmasını temin eder. α değerinin yüksek

olması feromonun yoğun olduğu yolların seçilme olasılığını arttırırken tesadüfiliği azaltmaktadır.

β: Problemde görünürlük değerine gösterilen önem dercesi veya parametresi
β değeri yol uzunluklarının, bir sonraki noktanın seçimindeki etkisini belirlemektedir. β değeri arttıkça bir sonraki yolun seçiminde tesadüfilik artmaktadır. β 'nın düşük olması ise alternatif çözümlerin araştırılması ihtimalini azaltır.

t_{Ak}: Henüz seçilmemiş olan düğüm noktaları kümesi (kϵAk dır.)

Karıncalar bir sonraki seçimlerini yukarıdaki olasılık bağıntısına göre yaparlar. Problemdeki tüm düğüm noktaları gezildikten sonra bir tur veya iterasyon tamamlanmıştır. Bundan sonra aşağıdaki bağıntıya göre feromen iz miktarı güncellenir [62].

$$t_{ij}(t+n) = p.\,t_{ij}(t) + \Delta t_{ij}$$

p: t ile t+n zaman aralığında kaybolan feromen iz oranı (0 < p < 1)

Δt_{ij}: Karıncanın bir turu boyunca (i, j) köşe tercihinden ötürü, köşedeki feromen iz miktarı. Bu aşağıdaki formülle hesaplanır.

$$\Delta t_{ij} = \sum_{k=1}^{m} \Delta t_{ij}{}^{k}$$

m: Toplam karınca sayısı

$\Delta t_{ij}{}^{k}$: k. karıncanın (i, j) köşelerine bıraktığı feromen miktarıdır.

Aşağıdaki bağıntı ise her bir k adet karıncanın herhangi bir (i, j) köşesindeki feromen miktarına ne kadarlık katkı yapacağını gösterir.

$$\Delta t_{ij}{}^{k} = \frac{Q}{L_k}$$

Q: Sabit değer

L_k: k. karıncanın attığı tur sayısı

Eğer karınca (i, j) köşesini kullanmamışsa iz miktarı 0 dır.

Karınca koloni optimizasyonu, birçok kombinatoriyel optimizasyon problemlerinin çözümünde kullanılmıştır. Dorigo tarafından yapılan ilk çalışmada karınca sistemi algoritması tanıtılmış ve deneysel sonuçlar verilmiştir. Literatürde en fazla yayına gezgin satıcı probleminin farklı karınca algoritmaları ile çözümü şeklinde karşılaşılmaktadır. 1995 yılında Gambardella ve Dorigo [Bkz. A Reinforcement Learning Approach to the Traveling Salesman Problem., In Proceedings of the Eleventh International Conference on Machine Learning, 1995 s.252-260]

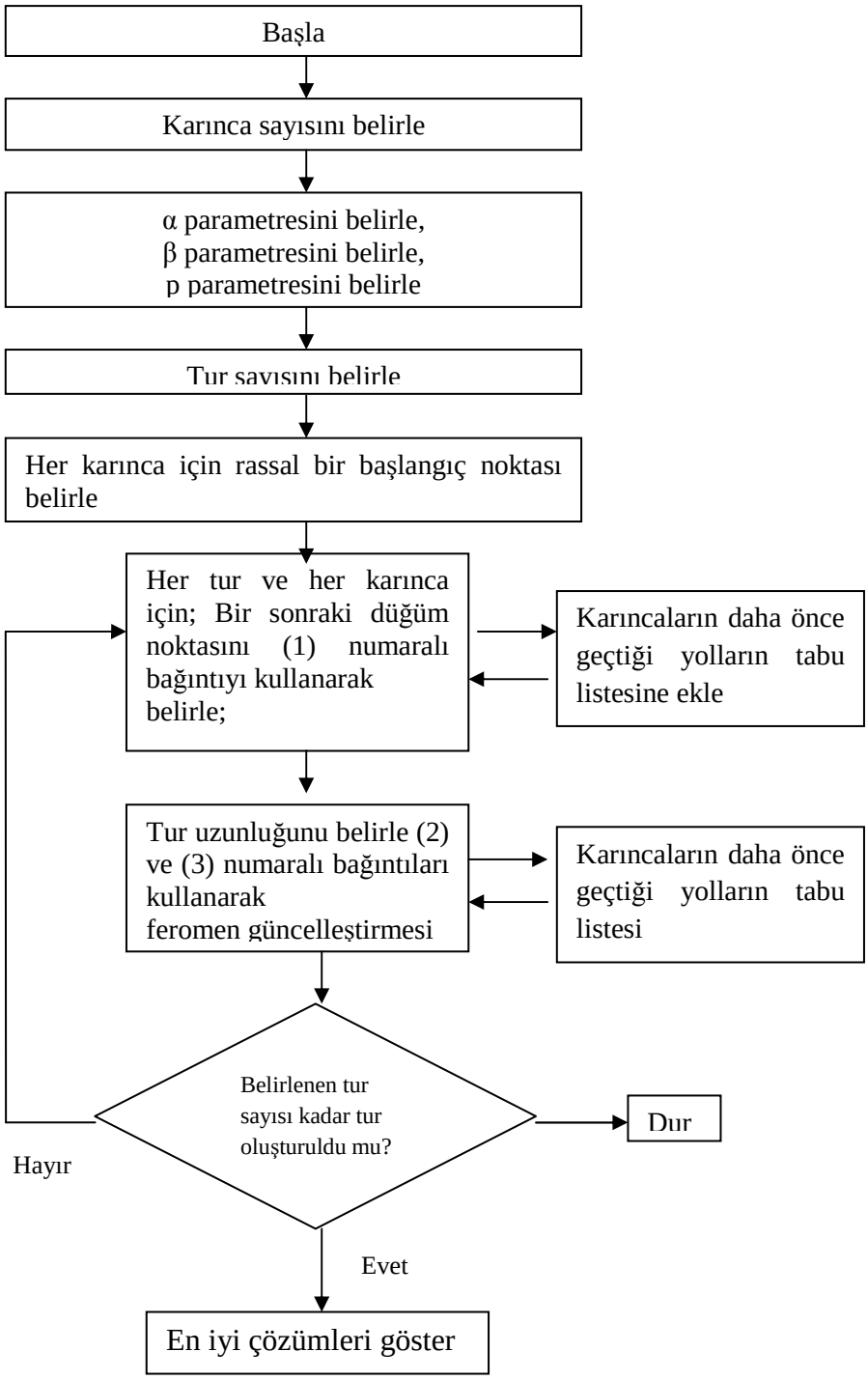
“Ant-Q” adında bir öğrenme algoritması ile, 1997 yılında Dorigo ve Gambardella karınca koloni sistemi ile, 1997 yılında Stützle ve Hoos [Bkz. The Max-Min Ant System and Local Search for the Traveling Salesman Problem, Proceedings of the IEEE International Conference on Evolutionary Computation (ICEC'97), 1997] max-min karınca sistemi ile ve yine 1997 yılında Bullnheimer ve arkadaşları mertebe temelli karınca sistemi ile bu probleme çözüm getirmişlerdir. [Bkz. BULLNHEIMER B., HARTL R.F., STRAUSS C., A New Rank Based Version of the Ant System: A Computational Study, Central European Journal for Operations Research and Economics, 1997] Karınca algoritmaları ile yapılan deneylerde en iyi sonuçlar max-min karınca sistemi ile elde edilmiştir. Gezgin satıcı probleminden sonra en fazla çalışma karesel atama probleminde yapılmıştır. Karesel atama problemine çözüm getiren tüm bu çalışmalarda karınca sistemi, karesel atama probleminin kısıtlarına ve şartlarına uydurularak yeni algoritmalar türetilmiştir.

Bu problemler dışında literatürde, rotalama problemi, grafik boyama, atölye tipi çizelgeleme problemi, maksimal kısıt sağlama problemi, klavye düzenleme problemi, tek makine çizelgeleme problemleri, genel kesikli optimizasyon problemleri ile ilgili yayınlarla karşılaşılmaktadır. Diğer algoritmalardan farklı olarak, Kawamura ve arkadaşları çok sayıda koloni içeren bir karınca algoritması tanıtmışlardır. Bu çalışmada aynı anda çok sayıda karınca kolonisi gezgin satıcı probleminin çözümünde kullanılmıştır. Karınca sistemi ile yapılan karşılaştırılmalı testlerde %2 ile %4 arasında daha iyi sonuçlar elde edilmiştir. [Bkz. KAWAMURA H., YAMAMOTO M., Multiple Ant Colonies Algorithm Based On Colony Level Interactions, IEICE Trans. Fundamentals, 2000]

White ve arkadaşları 2000 yılında karınca sistemi ile genetik algoritmaları birleştirerek hibrid bir uygulama yapmışlardır. Yapılan deneyler sonucunda hibrid algoritmanın daha başarılı sonuçlar verdiği görülmüştür. [Bkz. WHITE T., PAGUREK B., OPPACHER F., ASGA: Improving the Ant System By Integration With Genetic Algorithms, Systems and Computer Engineering, Carleton University Pres, 2000] Literatürde, gerçek bir endüstriyel problemin çözümüne karınca algoritmaları ile yaklaşan tek çalışma Gagne ve arkadaşları tarafından 2001 yılında yapılmıştır. Bu çalışmada arıza zamanlarının ve tamir sürelerinin dikkate alındığı çok amaçlı bir çizelgeleme problemi ele alınmıştır. [Bkz. GAGNE C., GRAVEL M., PRICE W., A Look-Ahead Edition to the Ant Colony Optimization Metaheuristic And Its Application To An Industrial Scheduling Problem, 4th Metaheuristics International Conference, 2001]

3.7.2. Karınca koloni algoritmasının akışı

Karınca sayısını belirle,



Şekil 3.49. Karınca Koloni Algoritmasının Akışı

3.8. Tabu Arama Optimizasyon Algortiması (Tabu Search)

Tabu arama (TA), optimizasyon problemleri için geliştirilmiş son zeki optimizasyon tekniklerinden birisidir. TA' nın sürekli optimizasyon problemlerinin çözümüne katkıları, diğer zeki tekniklerle kıyaslandığında hala oldukça sınırlıdır [64].

TA çözümde bir değişiklik yapmayı kabul etmeden önce mevcut çözümlerin komşularının, başka bir ifadeyle yakın çözümlerin bulunduğu kümede arama yapar. Çözüm uzayında yapılabilecek mümkün hareketler kümesinin üretilmesi ve bunlardan birisinin kabul edilmesi iterasyonlar boyunca devam eder [63]. TA' nın temelleri, uygunluk sınırlarını veya doğal olarak bariyer vazifesi yapan lokal optimalliği aşmak ve sistematik olarak kısıtları zorlayarak yasak alanlarda araştırma yapmaya izin vermek için dizayn edilmiş metotlara dayanır. Bu tür prosedürlerin ilk örnekleri, sistematik olarak uygunluk koşullarını zorlayan vekil kısıt metotlarını (surrogate constraint methods) ve düzlem kesme yaklaşımlarını (cutting plane approaches) temel alan sezgisel yaklaşımları kapsamaktaydı. TA' nın modern biçimi Glover ile şekillenmiştir [63].

Glover' dan bu yana TA alanında çok sayıda çalışma ortaya konmuştur. Bu çalışmaların büyük çoğunluğu optimizasyon problemleri ile ilgilidir. Battiti ve Tecchiolli "sürekli tepkili Tabu Arama" (Continuous Reactive Tabu Search) metodu ortaya koymuşlardır. Optimuma en yakın çözümleri barındıran kutucukların yerini belirlemeye ve daha sonra lokal arama için bu kutucuklar içerisinden başlangıç noktası seçmeye çalışılmıştır. (Bkz. BATTITI, R., TECCHIOLLI, G., The Continuous Reactive Tabu Search: Blending Combinatorial Optimization And Stochastic Search For Global Optimization, Annals of Operations Research, 1996, s.153-188)

TA lokal optimalliğin ötesindeki çözüm uzayını araştırmak için lokal sezgisel arama prosedürü sunan bir tekniktir. TA' nın ana parçalarından birisi uyarlanabilir hafızasıdır. Böylece daha esnek bir arama davranışı ortaya çıkar. Bu sebeple hafıza temelli stratejiler TA yaklaşımlarının ayırt edici özellikleridir.

3.8.1. TA’ nın en temel elemanları ve işleyişi

Başlangıç çözümünün oluşturulması

Başlangıç çözümü genelde rassal olarak elde edilir. Fakat ilgilenilen problem için geliştirilmiş olan bir algortimadan yararlanılarak ta başlangıç çözümü elde etmek mümkündür.

Hareket mekanizması

Bu mekanizma mevcut çözümde değişiklik yapılarak yeni çözümlerin elde edilmesini sağlar. TA’ nın etkin çalışması bakımından önemlidir. Problem yapısına bağlı olarak uygun bir şekilde seçilmelidir.

Stratejiler

TA’ nın belirli 3 temel hareket stratejisi vardır.

a) Yasaklama Stratejisi: Tabu listesine neyin gireceğini kontrol eder.
b) Serbest Bırakma Stratejisi: Tabu listesinden kimin ne zaman çıkacağını kontrol eder.
c) Kısa zamanlı strateji: Yasaklama stratejisi ile serbest bırakma stratejisi arasında etkileşimi sağlar.

Hafıza

Arama boyunca ortaya çıkan durumlar hafızaya kaydedilir. Yapılmasına izin verilmeyen hareketler tabu olarak adlandırılır ve esnek hafızası içinde tabu listesi adı altında kaydedilir.

Tabu yıkma kriterleri

Tabunun ortadan kalkabileceği durumları ifade eder. Mevcut durumdan daha iyi bir sonuç verecek tabu hareketinin yapılmasına izin verilmesi en genel yıkma kriteridir.

Durdurma koşulu

Durdurmayı sağlayan koşullar aşağıdaki gibidir.

Belirli adım sayısına ulaşılmaması
Arzu edilen çözüm değerinin yakalanması
Seçilen bir komşu çözümün komşusunu olmaması
Algoritmanın belirli bir noktadan sonra sonuç üretememesi

BÖLÜM 4. SONUÇLAR ve ÖNERİLER

4.1. Sonuçlar

Tez çalışmasının amacı klasik optimizasyon tekniklerini değerlendirerek, zeki optimizasyon tekniklerini inceleyip birbirleri ile arasındaki mukayese durumunu belirlemektir. Bu kapsamda önem teşkil eden nokta, zeki optimizasyon tekniklerinin ortaya çıkış sebepleri veya klasik optimizasyona göre avantajlarının neler olduğunu göstermektir.

Zeki optimizasyon tekniklerinin yukarıda problem örneklerinde de gösterildiği gibi klasik tekniklere göre avantajları şunlardır:

4.1.1. Sonuçların hızlı alınması

Zeki optimizasyon tekniklerinde sonuçlar daha hızlı üretilir. Çözüm uzayı büyük olan problemlerde klasik tekniklerde tüm olası çözümleri değerlendirmek oldukça uzun bir zaman gerektirir. Zeki optimizasyon tekniklerinde ise bu zaman dilimi oldukça kısadır ve tüm olası çözümler yeterli derecede değerlendirilir.

4.1.2. Uygulama kolaylığı

Zeki optimizasyon tekniklerinde kullanılan bulgusal kurallar ile karar verici veya kullanıcı için önemli bir nokta olan anahtar parametrelerin seçilen hareketler üzerindeki etkisinin nasıl olduğu, kolay bir şekilde anlaşılmaktadır. Burada Bölüm 3.5 te (Bkz. Sayfa 202) çözülen problem örnek gösterilebilir. Sürü optimizasyon algoritması herhangi bir diferansiyel denklem çözümü ve gradyan hesabı

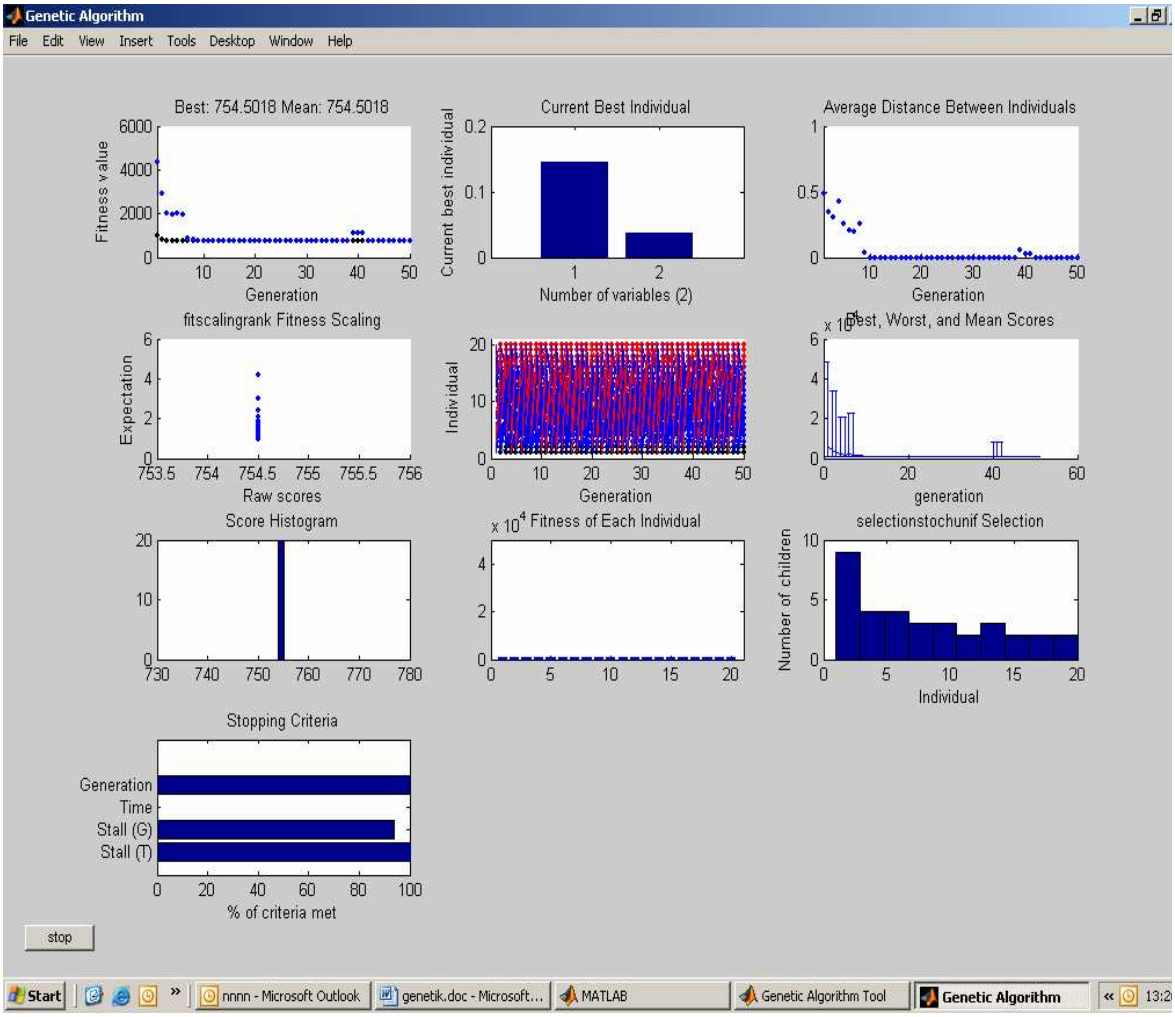
gerektirmediğinden çok boyutlu problemlere oldukça hızlı ve yeterli sonuçlar verdiği görülmektedir. Bu nedenle gradyan tabanlı klasik öğrenme algoritmaları yerine SO' nun kullanılabileceği söylenebilir.

4.1.3. Klasik tekniklere göre daha iyi sonuçların alınması

Zeki optimizasyon tekniğinde, genellikle bir bulgusal çözüm mevcut çözüme göre kullanıcıya daha tatminkar sonuçlar üretir.

Bu noktadan hareketle tez çalışmasının bölüm 3.3. (Bkz. Sayfa 124) kısmında yer alan fonksiyonun minimumlaştırılması örnek probleminin genetik algoritma ile çözüm çıktılarını göstermek gerekmektedir.

$f(x) = 5x_1{}^2 + 2x_2{}^2 + 2x_3{}^2 + 2x_1x_2 + 2x_2x_3 - 2x_1x_3 - 6x_3$ fonksiyonunun minimizasyonu genetik algoritma kullanarak çözümlendiğinde aşağıdaki sonuçlar elde edilir.



Şekil 4.1 GA İle Çözümün Ekran Çıktıları

Şekil 4.1 de gösterilen ekran çıktılarının anlamı açıklanacak olursa;

Birinci tablo optimimum çözümün ne olduğunu gösterir, burada optimum sonuç 754.5018'dir.

İkinci tablo optimum sonuca ulaşmak için karar değişkenlerinin ne olması gerektiğini ifade eder, yine bu koşumda 0,14567, 0,03643 olarak bulunmuştur.

Üçüncü tablo ise karar değişkenleri arasındaki ortalama uzaklığın ifadesidir, şekilden de anlaşılacağı üzere 10. nesile kadar karar değişkenleri arasındaki uzaklık çok değişmiş, 10. nesilden sonra düzene girmişti, daha sonra 40. nesil civarı biraz

değişim yaşanmıştır, bu değişimin rassal değişkenlerden dolayı meydana gelmiş olması muhtemeldir.

Dördüncü tablo, uyum fonksiyonunun değerlerinin sıralamasının bir göstergesidir, buradan da görüleceği gibi, uyum fonksiyonunun her bir nesilde aldığı değerlerin ağırlığı alındığında 754.5 etrafında yoğunlaşmaktadır.

Beşinci tablo, karar değişkenlerinin her bir nesilde nasıl bir seyir izlediğinin göstergesidir.

Altıncı tabloda, her bir jenerasyondaki en iyi, en kötü ve ortalama skor değerleri yer almaktadır.

Yedinci tablo, altıncı tablonun bir histogramıdır. Buradan da yine optimum sonucun 754.5 olduğu gözlemlenmektedir.

Sekizinci tablo, her bir karar değişkeninin uyumluluğunu göstermektedir.

Dokuzuncu tablo, karar değişkenleri için üretilen birey sayısını göstermektedir.

Onuncu tablo ise durma kriterlerinin % ne kadar karşılandığının bir ifadesidir.

Bu avantaj konusundaki diğer bir örnek, Bölüm 3.4. te gösterilen maliyet minimizasyonu problemi çözümüdür. Bilindiği gibi güncel hayatta ilgili veri ve model parametreleri farklılık gösterebilecektir. Önerilen yaklaşımda, maliyet minimizasyon hedefini sağlayabilmek için karar vericiye bir tatmin derecesi belirlenir. Bu dereceyi λ değeri gerçekleştirir, Yani elde edilen sonuçlar mevcut değişkenler ve bulanıklaştırdığımız kısıtların sağ taraf değişkenleri ile sağlanmıştır. Kıt kaynakların arttırılması ile amaç daha iyileştirebilir, λ düzeyi arttırılabilir. Bu nedenle amacı karşılayan çözüm elde edilinceye kadar, halihazırda olan belirsiz verilerle, ilgili model parametreleri sürekli değiştirilmelidir. Bulanık optimizasyon karar vericiye böyle bir imkan sunar. Doğrusal optimizasyonda ise güncel hayatta yaşanan belirsizliğin etkileri görülemez.

4.1.4. Değişken sayısının azaltılması

Çözüm uzayı ve değişken sayısı arasında sıkı bir korelasyon vardır. Değişken sayısının artması çözüm uzayını arttırır. Optimizasyon problemlerinde amaç olabildiğince değişken sayısı azaltılmaya çalışılarak problem çözümü gerçekleştirilmektir. Zeki optimizasyon tekniklerinde problemin değişkenleri, ortak değişkenler ile oluşturulup bir model kurulduğundan değişken sayısı azaltılmış olur.

4.1.5. Dayanıklılık

Zeki optimizasyon tekniklerinde problemin karakteristikleri ile ilgili ufak değişiklikler verinin kalitesi üzerinde hassas değişimler meydana getirmez, klasik optimizasyon tekniklerinde bu durum böyle değildir, burada elde edilen optimal çözümler veri değişimlerine karşı oldukça hassastır, ufak bir değişim optimal çözümü elde etmek için problemin yeniden çözümlenmesini gerektirecektir, bu oldukça zaman alıcıdır. Diğer taraftan zeki optimizasyon tekniklerinde kullanılan bulgusallar, çözüm uzayını çeşitli kısımlara ayırdıkları için, değişim etki alanını sınırlı bir düzeye indirger, hesaplama lokal bir alanda olur, bu nedenle klasik teknikere göre daha hızlıdır.

4.1.6. Kaynakları etkili şekilde kullanmak

Klasik optimizasyon teknikleri ile karmaşık problemlerin çözümü için gelişmiş bilgisayarlar gerekmektedir. Büyük miktarlarda donanım sağlanarak üretim programlarının şekillenmesi, yeni bir tesis tasarımının yapılması veya bir sistemin oluşturulması akılcı bir seçim yolu değildir. Bu nedenle uygun bir bulgusal fonksiyon kullanarak daha az cpu zamanı harcanıp ve daha az donanım kullanarak etkin bir kullanım yapılmış olur.

4.1.7. Birlikte kullanım avantajı

Klasik optimizasyon teknikleri zeki optimizasyon teknikleri ile bütünleşik kullanıldığı zaman daha verimli sonuçlar oluşur. Bütünleşik kullanım üç farklı amaç

için olur. Birincisi; daha iyi bir başlangıç çözümü oluşturmak, ikincisi; çözüm uzayını parçalara ayırarak alanı küçültüp işlemleri hızlandırmak, üçüncüsü ise; bulgusal fonksiyonları arama uzayında yol göstermek amacıyla kullanmaktır.

4.2. Öneriler

İşlemsel zeka (computational intelligence) doğası itibari ile güçlü bir araştırma ve optimizasyon mekanizmasıdır. Parametre sayısının çok olduğu ve verilerin normal toplanmadığı birçok kombinatoriyel optimizasyon (combinatorial optimization) ve mühendislik problemlerinin çözümünde bilgisayarlar kullanılmaktadır. Bu tür optimizasyon problemlerin çözümünde klasik optimizasyon teknikleri yetersiz kalmakla beraber son yıllarda zeki optimizasyon tekniklerinin karmaşık optimizasyon problemlerinin çözümünde başarılı sonuçlar verdiği görülmektedir.

Zeki optimizasyon tekniklerinin kullanılmasındaki amaç düşük maliyet ve zamandan tasarruf sağlayarak en uygun çözüme ulaşmaktır. Geliştirilen teknikler doğrultusunda doğru analiz yapabilme ve anında karar verme gibi insana özgü olan yetileri kullanarak karar veren veya tavsiyelerde bulunan sistemlerin geliştirilmesi ile daha hızlı ve gerçekçi çözümler elde edilecektir.

Bu çalışmada klasik optimizasyon tekniklerinden ve zeka optimizasyon tekniklerinden bahsedilerek literatürde bulunan bazı örnekler verilerek sınıflandırma yapılmıştır. Uygulamalar klasik optimizasyon tekniklerinin kullanılmasının yetersiz olduğu durumlarda yeni arayışları beraberinde getirmiş ve araştırmacılar bu yönde çalışmalarını sürdürmüşlerdir. Zeki optimizasyon teknikleri ise bu çalışmaların bir neticesi olarak ortaya atılmıştır.

Karmaşık problemlerin çözümüne yeni teknikler geliştiren yapay zeka, hem kullanım teknikleri açısından hem de optimizasyon algoritmaları açısından araştırmalarda sık başvurulan bir uygulama alanı olmuştur. Organik olmayan bir sistem olan yapay zeka, insan düşünüş ve davranış biçiminin bir benzetim modelidir. Bu sebeple, geliştirilen zeki teknikler de, insanın ve çevrenin davranışlarını modelleyerek problem çözmeyi amaçlamıştır, bunlara örnek göstermek gerekirse, bulanık mantık,

problem çözümünde dilsel ifadeleri kullanmış, yapay sinir ağları beyin nöronlarını kullanmış, sürü optimizasyonu canlının çevresiyle ve diğer bireylerle olan etkileşimi ve kolektif (ortak) davranışlarını, genetik algoritmalar ise doğal olayları modelleyen bir optimizasyon tekniği olarak ortaya çıkmıştır.

Genel olarak klasik optimizasyon tekniklerinden farklı olarak geliştirilen zeki optimizasyon algoritmaları, uygulamalarda olumlu sonuçlar vermesine karşın, dezavantajları olduğu da görülmektedir. Araştırmacılar bu dezavantajları giderecek yeni çalışmalar yapmaktadırlar. Optimizasyon problemlerinde kullanılan klasik tekniklerde amaç fonksiyonunun ve problem parametrelerinin matematiksel olarak ifade edilmesine karşın zeki optimizasyon algoritmalarında bu tür bir kısıt bulunmamaktadır. Bu da problem çözümü için bir avantaj sağlamaktadır. Araştırmacılar bu esnekliği kullanarak en uygun çözüme hızlı ve kabul edilebilir ölçüde ulaşabilmektedirler. Hesaplamalarda; algoritmanın klasik yöntemlere göre işlem yükü daha azdır. Ancak çözülecek problemdeki parametre sayısına da bağlı olmakla birlikte, algoritmada kullanılan parçacık sayısı, bellek miktarı gereksinimini artırabilmektedir. Optimal programlama becerisi ile bu sorunun aşılabileceği düşünülmektedir.

Ayrıca birden fazla zeki optimizasyon yönteminin birlikte kullanımı diğer bir ifade ile melez sistemlerin kullanımı da performansı arttırabilecek yaklaşımlardan birisidir. Örnek olarak, problem çözümlenirken, sistemin modeli çıkarılırken yapay sinir ağlarının öğrenme özelliğinin yanında bulanık mantığının karar verme özelliğinin de birlikte kullanıldığı uyarlanabilir, problem çözümü için uygun (adaptif) neural bulanık sistemler kullanmak çözüm performansını arttırır.

Gerçek dünyada olaylar ve olayların arkasındaki değişik birçok faktör vardır. Bunların birbirleri ile olan ilişkilerini ve birbirleri üzerindeki etkilerini gerçek hayatta bilmek zordur. YSA bu ilişkileri örneklerden öğrenir. Kullanıcını bu ilişkiyi bilmesi ve ağa söylenmesi beklenmediğinden, problem çözümde etkin bir verimlilik sağlanır, fakat YSA' nın performansın geliştirmek için katman sayısı, katmanlardaki nöron sayısı, öğrenme ve momentum katsayıları ve eğitim algoritmasının seçimi

büyük önem taşır, bu parametrelerin seçimi için diğer zeki tekniklerden faydalanmak bu gelişim için önemlidir.

Sürü Optimizasyonu son yıllarda kullanılan zeki tekniklerden biri olarak, karmaşık denklem takımı içeren lineer olmayan problemlerde başarı ile kullanılmaktadır. Klasik optimizasyon problemlerinden en önemli farkı türev kullanmamasıdır. Bu da sonuca daha kısa sürede ulaşmasını sağlamaktadır. Daha karmaşık problemlerde bu tekniğin geliştirilerek kullanılması problem çözümlemelerinde etkinliği arttıran bir faktör olacaktır.

Ölçeğin büyük olduğu, çok değişkenli problemlerde modern kontrol teorisi ile nonlineer performans fonksiyonları oluşturularak, klasik gerçek problemler; özellikle bulanık optimal kontrol ve diğer tekniklerle, bulanık kaotik problemler karşılaştırılarak, örnek uzayını küçültüp en optimal çözüme ulaşmak için araştırmalar yapılmalıdır.

KAYNAKLAR

[1] SARKER, Ruhul A, Optimization Modelling, A Practical Approach, CRC press. 2007.

[2] ELSTER K. H., Modern Mathematical Methods of Optimization, Vch Pub 1993.

[3] PANOS Y. P., and DOUGLASS J. W., Principles of Optimal Design : Modeling and Computation, Cambridge University Press. 2000.

[4] WAYNE L. Winston, Operation Research Applications And Algorithms, Third Edititon, Thomson Publishing, USA, 1994.

[5] TAHA H, Yöneylem Araştırması, 6. Baskı, Çeviren: S.Alp Baray, Sakir Esnaf, İstanbul, 2000.

[6] DROR, M., P. TRUDEAU AND S.P. LADANY. Network Models For Seat Allocation On Flights. Transportation Research,. The University of Michigan, Ann Arbor. 1988, s.239-250.

[7] THOMAS H. C., LEISERSON C. E., RIVEST R. L., and STEIN C., Introduction to Algorithms, Second Edition. MIT Press and McGraw-Hill, 2001, s.770–821.

[8] AHUJA, R.K., Algorithms for the minimax transportation problem. Naval Research Logistics Quarterly, 1986, s.725-740.

[9] FEDERGRUEN, A., AND H. GROENEVELT.. Preemptive scheduling on uniform machines by network flow techniques. Management Science, 1986, s.341-349.

[10] GLOVER, F., AND J. ROGOZINSKI. Resort development: A Network Related Model For Optimizing Sites And Visits. Journal of Leisure Research, 1982, s.235-247.

[11] YANG X. S., Introduction to Mathematical Optimization: From Linear Programming to Metaheuristics, Cambridge Int. Science Publishing, 2008.

[12] BRYSON, A. E., Dynamic Optimization, Addison Wesaley Longman Inc, 1999.

[13] FU, M. C. "Optimization for Simulation: Theory vs. Practice". INFORMS Journal on Computing 2002, s.192–227.

[14] NOCEDAL, J., WRIGHT S. J., Numerical Optimization second edition, Springer Verlag, 2006.

[15] ANDERSON D. R., SWEENEY D. J. and WILLIAMS T.A.: An. Introduction to Management Science; Approaches to Decision Making, West Publishing, 2000, s.827-828.

[16] OZDEMİR, N., EVIRGEN, F., YAMAN R. Yaman., A New Solution Technique for MIMO System of Unconstrained Quadratic Optimization Problems using State-Space Approach, Euro XXI 2006, 21st European Conference on Operational Research, University of Iceland, Reykjavik, July 2-5, 2006.

[17] EVIRGEN, F., OZDEMIR N A System of ODEs for Nonlinear Programming Problems, The 20th International Congress of Jangjeon Mathematical Society, Bursa, Turkey, August 21-23, 2008.

[18] NARAYAN V., Effective Maintenance Management: Risk and Reliability Strategies for Optimizing Performance Industrial Press 2004, s.127-159.

[19] SHARKAWI E. M., LEE Y. K., Modern Optimization Techniques Theory And Applications To Power Systems John Willey, 2008, s.4-7

[20] ENGELBRECHT A. F., Computational Intelligence An Introduction, John Willey 2007.

[21] SARAVANAN. R., Manufacturing Optimization: Through İntelligent Techniques, CRC Press, 2006.

[22] ÖZTEMEL E., Yapay Sinir Ağları Papatya Yayınları 2008.

[24] MASON A., RYAN D., PANTON D.,Integrated Simulation, Heuristic And Optimization Approaches To Staff Scheduling, Operations Research 1998, s.161-175.

[25] LUGER., G. F., Artificial Intelligence: Structures And Strategies For Complex Problem Solving, Addison-Wesley, 2002.

[26] H. DEMUTH, M., BEALE, M. HAGAN, "Neural Network Toolbox, For Use with MATLAB", 2006.

[27] BASHIR, H. A. and THOMSON, V., Models For Estimating Design Effort And Time, Design Studies Vol 22 No 2 March 2001.

[28] FINNIE, G.R., WITTIG, G.E. and DESHARNAIS, J. M., A Comparison of Software Effort Estimation Techniques: Using Function Points with Neural Networks, Case-Based Reasoning and Regression Models, Journal of Systems Software, v 39, Elsevier Science Inc. 1997, p: 281-289.

[29] HUANG, X., HU D., REN J., and Capretz, L.F., A Soft Computing Framework For Software Effort Estimation", Soft Computing Springer-Verlag., 2006, s.170–177.

[30] HOLLAND. J.H., Adaptation In Natural And Artificial Systems, University Of Michigan Press, An Arbour, MI, 1975.

[31] ENGELBRECHT A. F., Computational Intelligence An Introduction, John Willey 2007, s.127-185.

[32] CHANG, C. K., CHRISTENSEN, M. J. and ZHANG, T., Genetic Algorithms for Project Management, Annals of Software Engineering, Kluwer Academic Publishers 2001, s.107–139.

[33] BRIAND, L. C., FENG, J. and LABICHE, Y., Using Genetic Algorithms and Coupling Measures to Devise Optimal Integration Test Orders, ACM - SEKE '02, July 2002, s.15-19. Ischia, Italy.

[34] EIBEN A.E., AND SMITH J.E., Introduction To Evolutionary Computing, Springer 2003.

[35] GOLDBERG, D.E., Genetic Algorithms in Search, Optimization, and Machine Learning. Addison-Wesely, Reading MA 1989.

[36] CICIRELLO, V.A., SMITH, S. F., "Modelling GA Performance for Control Parameter Optimization", Genetic and Evolutionary Computation Conferenc, Las Vegas, Nevada, USA, July 8-12. 2000.

[37] BURGESS, C.J. and LEFLEY, M., Can Genetic Programming İmprove Software Effort ? A Comparative Evaluation, Information and Software Technology, Elsevier., 2001, s.863-873.

[38] MATLAB, 2002, The Language of Technical Computing, Version Release 13, Copyright 1984-2002, The MathWorks,Inc.

[39] KOSKO, B. and ISAKA S. Fuzzy Logic. Scientific American, 1993, s.76-81.

[40] ZHANG, L, and CAI K.Y., International Journal of Intelligent Systems; Optimal Fuzzy Reasoning And İts Robustness Analysis vol:19, 2004, s.1033-1049.

[41] WU, D.R., TAN, W.W., Computationally Efficient Type-Reduction Strategies For Fuzzy Logic Controller. In: Proceedings of Fuzzy-IEEE, USA 2005, s.353-358.

[42] TRIANTAPHYLLOU E., Mult-Criteria Decision Making Methods: A Comparative Study, Kluver Academic Publishers, Dordrecht, 2000.

[43] ENGELBRECHT A. F., Computational Intelligence An Introduction, John Willey 2007, s.475-478.

[44] JANG, J. S., SUN C. T., and MIZUTANI E., Neuro-Fuzzy and Soft Computing: A. Computational Approach to Learning and Machine Intelligence, Prentice Hall, 2002.

[45] XU, Z. and KHOSHGOFTAAR, T.M., Identification Of Fuzzy Models Of Software Cost Estimation, Fuzzy Sets and Systems, , Elsevier, 2004, s.141-163.

[46] ZIMMERMANN H. J., Fuzzy Sets, Decision Making And Expert Systems, Kluver Academic Publishers, Dordrecht, 1987.

[47] CHANAS, S. The Use Of Parametric Programming in Fuzzy Linear Programming, Fuzzy Sets and Systems, 1985, Vol.11, Issue.3

[48] WERNERS B. A Coordination Method For Fuzzy Multi-Objective Optimization Of System Reliability: An İnteractive Fuzzy Programming System, Fuzzy Sets and systems, 1987, s.131–147.

[49] VERDEGAY, J. L., “Fuzzy Mathematical Programming in:M.Grupta and

E.Sanchez", Fuzzy Information and Decision Processes, 1982, s.231-237.

[50] WHANG, R, LIANG, T., 'Applying Possibilistic Linear Programming To Aggregate Production Planning', Int j. Production Economics, 2004, s.328-341.

[51] SADEGNİ M., HOSSEİNİ M., Energy Supply Planning In Iran By Using Fuzzy Linear Programming Approach (Regarding Uncertainties Of Investment Costs), Energy policy, 2005

[52] GASIMOV N.R., YENILMEZ K, Solving Fuzzy Linear Programming Problems with Linear Membership Functions, Tübitak 2002.

[53] GUU, SY-MİNG, WU YAN-KUEN, Two-Phase Approach For Solving The Fuzzy Linearprogramming Problems, Fuzzy Sets and Systems, Vol.101, Issue.1, 1996.

[54] KENNEDY, J. ve EBERHART, R. C. " Particle Swarm Optimization ", Proc. IEEE Conferans On Neural Networks, Vol:4, IEEE Service Center, Piscataway, NJ, 1995. s.1942-1946.

[55] ZHAO F., REN Z., YU D., YANG Y., "Application of An Improved Particle Swarm Optimization Algorithm for Neural Network Training", 0-7803-9422-04/05/2005 IEEE

[56] JUANG C. F., LU C. " Load-Frequency Control By Hybrid Evolutionary Fuzzy Pı Controller ", IEEE Vol. 153, No:2, March 2006, s.56

[57] GHOSHAL, S. P. " Optimizations Of Pıd Gains By Particle Swarm Optimizations In Fuzzy Based Automatic Generation Control", Electric Power Systems Research, Volume 72, Issue 3, 2004, s.203-212.

[58] AWAD H. A., " A Novel Particle Swarm-Based Fuzzy Control Scheme", IEEE International Conference on Fuzzy Systems, Canada July 16-21, 2006.

[59] SHI Y. And EBERHART, R. C., " Parameter Selection In Particle Swarm Optimization", Evolutionary Programming VII, Proc. Springer-Verlag, New York, 1998, s: 591-600.

[60] SHI, Y. and EBERHART, R. C. " A Modified Particle Swarm Optimizer", Proceedings of the IEEE International Conference on Evolutionary Computation IEEE Press, Piscataway, NJ, 1998, s.69-73

[61] DORIGO M., MANİEZZO V., Colorni A., The ant system: optimization by a colony of cooperating agents, IEEE Transactions on Systems, Man, and Cybernetics–Part, 1996.

[62] COLORNI A., DORIGO M., MANIEZZO V., Distributed Optimization by Ant Colonies. Proceedings of the First European Conference on Artificial Life, Paris, France, Elsevier Publishing, 1992.

[63] GLOVER F. And LAGUANA M., " Modern Heuristics Technics For Combinatorial Problems"; Tabu Search, Blackwell Scientific Publications, Oxford 1993, s.70-150.

[64] HEDAR, A., R., FUKUSHIMA, M., Tabu Search Directed By Direct Search Methods For Nonlinear Global Optimization, European Journal of Operational Research, 2006, s.329- 349.

[65] BATTITI, R., TECCHIOLLI, G., The Continuous Reactive Tabu Search: Blending Combinatorial Optimization And Stochastic Search For Global Optimization, Annals of Operations Research, 1996, s.153-188.

ÖZGEÇMİŞ

Ünal Atakan KAHRAMAN 1983 yılında Kayseri' nin Pınarbaşı ilçesinde doğdu. İlk, orta ve lise öğrenimini Sakarya'da tamlamıştır. 2004 yılında girdiği Sakarya Üniversitesi Mühendislik Fakültesi Endüstri Mühendisliği Bölümünü 2008 yılında dereceyle bitirdi. Aynı yıl yüksek lisans öğrenimine Sakarya Üniversitesi Fen Bilimleri Enstitüsü Endüstri Mühendisliği Anabilim Dalında başlamıştır. Halen Artvin Çoruh Üniversitesi Yöneylem Araştırması Anabilim Dalında Araştırma Görevlisi olarak çalışmaktadır.