DÜZCE ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

YÜKSEK LİSANS TEZİ

DOĞRUSAL PROGRAMLAMA PROBLEMLERİNİN META SEZGİSEL YÖNTEMLERLE ÇÖZÜMLENMESİ

Enver KÜÇÜKKÜLAHLI

ELEKTRİK EĞİTİMİ ANABİLİM DALI

MAYIS 2011
DÜZCE

DÜZCE ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

YÜKSEK LİSANS TEZİ

DOĞRUSAL PROGRAMLAMA PROBLEMLERİNİN META SEZGİSEL YÖNTEMLERLE ÇÖZÜMLENMESİ

Enver KÜÇÜKKÜLAHLI

ELEKTRİK EĞİTİMİ ANABİLİM DALI

MAYIS 2011
DÜZCE

Enver KÜÇÜKKÜLAHLI tarafından hazırlanan Doğrusal Programlama Problemlerinin Meta Sezgisel Yöntemlerle Çözümlenmesi adlı bu tezin Yüksek Lisans tezi olarak uygun olduğunu onaylarım.

Prof. Dr. İsmail ERCAN …….
Tez Danışmanı, Eğitim Programları ve Öğretim Anabilim Dalı

Bu çalışma, jürimiz tarafından oy birliği ile Elektrik Eğitimi Anabilim Dalında Yüksek Lisans tezi olarak kabul edilmiştir.

Prof. Dr. İsmail ERCAN …….
Eğitim Programları ve Öğretim Anabilim Dalı, Düzce Üniversitesi
Yrd. Doç. Dr. Pakize ERDOĞMUŞ …….
Bilgisayar Mühendisliği Anabilim Dalı, Düzce Üniversitesi
Yrd. Doç. Dr. Resul KARA …….
Bilgisayar Mühendisliği Anabilim Dalı, Düzce Üniversitesi
Tarih: 27/05/2011

Bu tez ile Düzce Üniversitesi Fen Bilimleri Enstitüsü Yönetim Kurulu Yüksek Lisans derecesini onamıştır.

Prof. Dr. Refik KARAGÜL …….
Fen Bilimleri Enstitüsü Müdürü

TEZ BİLDİRİMİ

Tez içindeki bütün bilgilerin etik davranış ve akademik kurallar çerçevesinde elde edilerek sunulduğunu, ayrıca tez yazım kurallarına uygun olarak hazırlanan bu çalışmada bana ait olmayan her türlü ifade ve bilginin kaynağına eksiksiz atıf yapıldığını bildiririm.

Enver KÜÇÜKKÜLAHLI

ÖNSÖZ

Yüksek lisans öğrenimim sırasında ve tez çalışmalarım boyunca gösterdiği her türlü destek ve yardımdan dolayı çok değerli hocalarım Prof. Dr. İsmail ERCAN ve Yrd. Doç. Dr. Pakize ERDOĞMUŞ'a en içten dileklerimle teşekkür ederim.

Bu çalışma boyunca yardımlarını esirgemeyen çalışma arkadaşlarıma teşekkürü borç bilirim.

Mayıs 2011 **Enver KÜÇÜKKÜLAHLI**

İÇİNDEKİLER **Sayfa**

ŞEKİL LİSTESİ

Sayfa

ÇİZELGE LİSTESİ

Sayfa

SEMBOL LİSTESİ

$\mathbf{a_{ij}}$: i. kısıt için a değişkeninin j.değer katsayısı
$\mathbf{b_i}$: i. kısıt için b değişkeninin değeri
$\mathbf{C_j}$: Sabit katsayı
$\mathbf{c_1, c_2}$: Ağırlıklandırma sabiti
dx : Sıçrama uzaklığı
$\mathbf{geniyi_d^t}$: t. iterasyona kadar komşu olarak belirlenen parçacıklar içersinde en iyi d değişken değeri
h : İçinde tabu sürelerini barındıran hareket vektörü
$\mathbf{h_{nd}^t}$: t. iterasyonda n. parçacığın d değişkeni için hızı
hx : Parçacığın x hızı
$\mathbf{h_x}$: Her hareketin tabu listesindeki karşılığı
$\mathbf{mevcut^{max}_{nd}}$: Parçacık pozisyonu için en yüksek değer
$\mathbf{mevcut_{nd}^t}$: t. iterasyonda n. parçacığın d değişkeninin değeri
$\mathbf{mevcutx_n}$: n karıncası için x konumu
$\mathbf{mevcuty_n}$: n karıncası için y konumu
n : Çözüm sayısı
$\mathbf{peniyi_{nd}^t}$: t. iterasyona kadar n. parçacığın en iyi d değişken değeri
$\mathbf{peniyi_n}$: Parçacığın en iyi uygunluk değeri
$\mathbf{peniyix_n}$: Parçacığın en iyi uygunluk değeri için x konumu
$\mathbf{peniyiy_n}$: Parçacığın en iyi uygunluk değeri için y konumu
$\mathbf{P_{tk}\{j'yi\ kabul\ et\}}$: Komşu Kabul etme olasılığı
rnd : [0-1] Aralığında rastgele değer
t : İterasyon numarası
$\mathbf{tabu\_başla_x}$: x hareketinin tabu olarak işaretlendiği iterasyon numarası
$\mathbf{tabu\_bitir_x}$: x hareketinin tabu olmaktan çıktığı iterasyon numarası
$\mathbf{Z_{max/min}}$: Amaç fonksiyonu (maximizasyon ya da minimizasyon)
x : Komşu seçme fonksiyonu (hareket)
$\mathbf{x_{0i}}$: i. değişkenin başlangıç noktası
$\mathbf{x_j}$: Karar değişkeni
$\mathbf{x_t^{best}}$: t. iterasyonda belirlenmiş en iyi çözüm
$\mathbf{\bar{x}_{initial}^{best}}$: Başlangıçta en iyi çözümü üreten karınca vektörü
$\mathbf{X_t^k}$: t. iterasyondaki k numaralı karıncayı ifade eden çözüm vektörü
$\mathbf{X_{initial}^k}$: k numaralı karınca için başlangıç vektörü
$\boldsymbol{\alpha}$: Problemin linprog ile çözümünden elde edilen asıl sonuç
$\boldsymbol{\alpha_p}$: p. problemin katsayısı
$\boldsymbol{\beta}$: Problemin meta sezgisel ile çözümünden elde edilen sonuç

$\boldsymbol{\Phi}$	: 0 ile 1 arasında rastgele üretilmiş sayı
$\boldsymbol{\omega}$	: Dinginlik katsayısı
$\boldsymbol{\emptyset_1, \emptyset_2, \emptyset_3}$	: [0-1] Aralığında rastgele değer
$\boldsymbol{\varepsilon}$	: Tabu süresi
$\boldsymbol{\varepsilon_{\text{bağıl}}}$	: Bağıl hata oranı
$\boldsymbol{\Psi(i)}$	: Hareket olarak adlandırılan komşu seçme fonksiyonu

KISALTMA LİSTESİ

BT - SA : Benzetilmiş Tavlama
EP : Evrimsel Programlama
ES : Evrim Stratejileri
GA : Genetik Algoritma
GP : Genetik Programlama
KKO - ACO : Karınca Koloni Optimizasyonu
LP : Doğrusal programlamadaki
P-KKO : Popülasyon Temelli Karınca Kolonisi
PSO : Parçacık Sürü Optimizasyonu
TA - TS : Tabu Arama

DOĞRUSAL PROGRAMLAMA PROBLEMLERİNİN META SEZGİSEL YÖNTEMLERLE ÇÖZÜMLENMESİ
(Yüksek Lisans Tezi)

Enver KÜÇÜKKÜLAHLI

DÜZCE ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ
Mayıs 2011

ÖZ

İnsanlık var olduğu zamandan beri aklını ve içinde yaşadığı dünyayı kullanarak günümüz teknolojisine ulaştı. Bu ilerlemeyi kaydetmesindeki en büyük etken ise hep "en iyi"yi aramasıdır. İyiyi arama işine "optimizasyon" adı verilmektedir. En iyiyi arama yolunda ilerleyen insanoğlu tıkandığında sezgilerine güvenerek çıkış yolu bulmaya çalıştı. Günümüzde ise en iyiyi bulma yolunda son derece karmaşık problemlerle karşılaşılabilmektedir. Günümüz şartlarında herhangi bir çözüm algoritması geliştirilemeyen ya da var olan algoritmalarla çözümü çok uzun zaman alan karmaşık problemlerde insanoğlu yine meta sezgisel yöntemlerden faydalanmaktadır. Son zamanlarda literatüre girmiş oldukça fazla meta sezgisel yöntem bulunmaktadır. Problem çözümlerinde, problemin yapısına göre; kullanılacak olan meta sezgisel yöntemin seçimi son derece önemlidir.

Bu tez çalışmasında, problem çözümlerinde kullanılan GA (Genetik Algoritma), KKO (Karınca Koloni Optimizasyonu), PSO (Parçacık Sürü Optimizasyonu), BT (Benzetilmiş Tavlama), TA (Tabu Arama) olmak üzere beş farklı meta sezgisel yöntem seçildi. Basitten karmaşığa 20 adet doğrusal programlama problemi bu meta sezgisel yöntemlerle, belirli koşullar altında sınandı ve sonuçlar değerlendirildi.

Elde edilen sonuçlara göre, yukarıda isimleri geçen beş algoritma arasından PSO (Parçacık Sürü Optimizasyonu) algoritmasının belirli koşullarda diğerlerine nazaran daha iyi sonuçlara ulaştığı gözlemlendi.

Bilim Kodu :

Anahtar Kelimeler : Doğrusal programlama, meta sezgisel, genetik algoritma, karınca kolonisi optimizasyonu, parçacık sürü optimizasyonu, benzetilmiş tavlama, tabu arama algoritması, optimizasyon

Sayfa Adedi : 83

Tez Yöneticisi : Prof. Dr. İsmail ERCAN

SOLVING OF LINEAR PROGRAMMING PROBLEMS WITH META HEURISTIC METHODS
(M.Sc. Thesis)

Enver KÜÇÜKKÜLAHLI

DUZCE UNIVERSITY

INSTITUTE OF SCIENCE AND TECHNOLOGY

May 2011

ABSTRACT

Humanity has reached today's technology, by using his mind and the world he lived in, since his existence. The most important factor contributing to this progress has always been his quest for "the best". The task of seeking the best is called optimization. When the road to advancement was blocked, humanity tried to find its way out, relying on intuition. Today, many complex problems can be encountered throughout this process. Mankind benefits from meta heuristic methods when an algorithm to solve the complex problems cannot be developed or employing the existing algorithm would take too much time. There are quite a lot of meta heuristic methods, which entered the literature recently. The selection of meta-heuristic method, which is to be used in the problem-solving process, regarding the structure of the problem, is extremely important.

In this thesis, five different meta-heuristics methods, being GA (Genetic Algorithm), ACO (Ant Colony Optimization), PSO (Particle Swarm Optimization), SA (Simulated Annealing) and TS (Tabu Search) are used to solve problem in problem-solving. 20 pieces of linear programming problem, from simple to complex, were tested and the results were evaluated with these meta-heuristic methods, under specific conditions. It is observed that PSO

(Particle Swarm Optimization) achieved better results than the others under specific conditions, according to the results of this study.

Science Code :
Key Words : Linear programming, meta heuristic, genetic algorithm, ant colony optimization, particle swarm optimization, simulated annealing, tabu search algorithm, optimization
Page Number : 83
Adviser : Prof. Dr. İsmail ERCAN

1. GİRİŞ

İnsanoğlu yaşamı boyunca eldeki kaynakları en verimli şekilde kullanmayı hedefler. İnsanoğlunun yüzyıllardır değişmeyen bu hedefi, günümüzde optimizasyon olarak tanımlanan olgunun tanımıyla birebir örtüşmektedir. Optimizasyon, bir sistemin minimum maliyetle maksimum verimi elde edebilmek için düzenlemesi işidir.

Optimizasyon, modelleme ve çözümleme olarak iki önemli bileşenden oluşur. Modelleme gerçek yaşamda karşılaşılan bir sorunun matematiksel olarak ifade edilmesi, çözümleme ise çeşitli araçlar kullanılarak bahsi geçen modelin en iyi çözümünün elde edilmesidir (Afacan, 2007).

Matematiksel olarak ifade edilen model; kısıtlayıcılarla sınırlandırılıyorsa kısıtlı, herhangi bir kısıtla sınırlandırılmıyorsa kısıtsız problemdir. Problem sadece bir dönem için çözdürülecekse statik, birden fazla dönem için çözdürülecekse dinamiktir. Model çalıştırılırken kesin parametreler kullanılıyorsa deterministik, parametreler olasılıklara bağlı olarak değişiklik gösteriyorsa stokastiktir. Model, amaç sayısına bakılarak çok amaçlı ya da tek amaçlı olarak nitelendirilir. Tüm karar değişkenleri tamsayı değerlerden oluşuyorsa kesikli optimizasyon problemi, pozitif reel değerlerden oluşuyorsa sürekli optimizasyon problemidir. Problemi oluşturan matematiksel model doğrusal olmayan bir denklem içeriyorsa doğrusal olmayan, doğrusal denklemlerden oluşuyorsa doğrusal programlama problemi olarak adlandırılır (Alataş, 2007).

Doğrusal programlamadaki doğrusal sözcüğü, modeldeki tüm fonksiyonların doğrusal olduğunu belirtirken programlama kelimesi planlama anlamında kullanılmaktadır (Özçiloğlu, 2009).

Doğrusal programlama problemlerinin çözümü için; simpleks yöntemi, üst ve alt sınırlandırma, ayrıştırılmış ve parametrik programlama gibi etkin hesaplama yöntemlerinin yanı sıra bu yöntemlerden tamamen farklı olan Karmarkar'ın iç nokta algoritması da kullanılabilmektedir. Bazı matematiksel modeller bilinen herhangi bir

optimizasyon algoritması ile çözülemeyecek derecede karmaşık olabilmektedir. Bu nedenle de optimum çözümü aramak son derece zahmetli olmaktadır. Böyle durumlarda optimum çözümü aramak yerine sezgisel yöntemler kullanılarak iyi bir çözümü arama yoluna gidilebilir (Taha, 2007).

Meta sezgisel yöntemler zor ve karmaşık optimizasyon problemlerinde iyi çözümlere ulaşabilmek için sezgisel yaklaşımları kullanan tasarımlar olarak adlandırılabilir (Kılıç, 2008). Klasik sezgisel yöntemlerden farklı olarak meta sezgisel yöntemler, probleme özgü tasarlanmadığı için problemin adımlarından ve özelliklerinden bağımsız olarak kullanılabilmektedirler. Bununla birlikte probleme özel tasarlanmış bir metot kadar iyi sonuç alınamayacağı ihtimali de bir dezavantaj oluşturmaktadır (Aydın, 2009).

Meta sezgisel algoritmaların önemli bir grubunu popülasyon temelli algoritmalar oluşturmaktadır. Bu gruptaki algoritmalar, probleme ait aday çözümleri barındıran bir popülasyon oluştururlar ve bu popülasyonu çeşitli operatörler yardımıyla geliştirerek yaklaşık çözümler üretirler. Popülasyon temelli algoritmaları, gelişime dayalı ve sürü zekâ temelli algoritmalar olarak iki alt gruba bölmek mümkündür. Genetik Algoritma (GA), Evrim Stratejileri (ES), Evrimsel Programlama (EP), Genetik Programlama (GP) gelişime dayalı algoritmalara; Karınca koloni algoritması (ACO), Parçacık Sürü Optimizasyon algoritması (PSO), Benzetilmiş Tavlama (SA), Tabu Araştırma (TS) sürü zekâ temelli algoritmalara örnek olarak gösterilebilir (Akay, 2009).

Bu bölümde popülasyon temelli algoritmalardan; Genetik Algoritma (GA), Karınca Kolonisi Optimizasyonu (ACO), Parçacık Sürü Optimizasyonu (PSO), Benzetilmiş Tavlama (SA) ve Tabu Arama (TS) algoritmaları incelenecektir.

1.1. POPÜLASYON TEMELLİ ALGORİTMALAR

1.1.1. Genetik Algoritma

Genetik Algoritmalar 1960'larda John Holland tarafından bulunmuş ve 1960 ve 1970'lerde Michigan Üniversitesinde Holland, öğrencileri ve meslektaşları tarafından geliştirilmiştir. Holland'ın GA metodu, bir popülasyondaki kromozomları bir tür doğal seleksiyon kullanarak, çaprazlama (crossover) ve mutasyon gibi genetik operatörler ile

yeni popülasyona almak için geliştirilmişti. Seçme operatörü ile o popülasyondaki iyi bireyler seçilmekte ve bu iyi bireyler kendi aralarında çaprazlanarak yeni bir birey oluşmaktaydı. Oluşturulan bu birey sonraki popülasyona aktarılmakta idi. John Holland, mutasyon, çaprazlama gibi kavramları literatüre tanıştıran kişi olması bakımından önemlidir. Holland, 1975 yılında çıkardığı "Adaptation in Natural and Artificial Systems" adlı kitabında genetik algoritmalar altında adaptasyon için teorik bir çerçeve sunmuştur (Mitchell, 1999, Gülcü, 2006).

1.1.2. Karınca Kolonisi Optimizasyonu

1989 yılında Goss ve diğ. (1989) laboratuar ortamında yetiştirdikleri karınca kolonilerini izleyerek çalışmalar gerçekleştirmişlerdir. Bu çalışmalardan bazıları ise kolonilerin incelenerek onlardan esinlenen yeni yöntem ve algoritmaların bulunması üzerine yoğunlaşmaktadır (Erdoğan, 2008). Karınca koloni optimizasyonu, ilk kez 1990'ların başında Dorigo ve meslektaşları tarafından zor kombinatoryal problemlerin çözümü için doğadan esinlenilmiş bir meta sezgisel olarak önerilmiştir (Dorigo ve Blumb, 2005). Koloniler halinde yaşayan böceklerde bireylerin davranışları bütünsel olarak kolonin hayatta kalmasına yöneliktir (Dorigo ve diğ., 1999). Bireysel olarak incelendiğinde bu böceklerin hareketleri anlamsız olarak nitelendirilebilirken kolonin tamamı gözlendiğinde en iyiyi arama yönünde olduğu söylenebilir. Karınca kolonilerinin en önemli ve ilginç davranışı ise yiyecek arama davranışları, özellikle de onların yiyecek ve yuvaları arasındaki en kısa yolu bulmalarıdır (Dorigo ve diğ., 1999).

1.1.3. Parçacık Sürü Optimizasyonu

Kennedy ve Eberhart (1995) tarafından, kuş ve balık sürülerinin iki boyutlu hareketlerinden esinlenilerek geliştirilmiş popülasyon tabanlı bir stokastik optimizasyon tekniğidir. Shi ve Eberhart (1998) tarafından optimizasyon modeline atalet ağırlığı eklenmiştir. Parçacık Sürü Optimizasyonu, sosyal yaşamın basitleştirilmiş bir simülasyonudur. PSO'da, parçacıların her biri bir çözüm adayını temsil eder. Bu parçacıklar bir optimizasyon probleminde çözüm uzayını araştırmak için kullanılırlar. Optimizasyon başlangıcında her bir parçacık rastgele ya da sezgisel olarak belirlenen bir konuma yerleştirilir ve sonraki adımlarda serbest harekete bırakılırlar. Her bir iterasyonda, her parçacık kendisinin ve çevresindekilerin uygunluğunu ölçer. Bu

ölçümler bir sonraki iterasyonda parçacığın yeni konumunu bulmasında kullanılır. Parçacık Sürü Optimizasyonunda parçacıklar kendilerinin geçmiş deneyimlerinden, uygunluk değerinin konumundan, komşularının geçmiş deneyimlerinden faydalanarak bir sosyal benzetim oluştururlar. Parçacığın, sürü içersindeki diğer parçacıklarla etkileşimi o parçacığın yönünü bulmasında etkili bir unsurdur. Bu şekilde sürü içersindeki parçacıklar zamanla uygunluk değeri iyi olan parçacıklara yaklaşırlar ve muhtemel en iyi uygunluk bölgesini daha sıkı tararlar.

1.1.4. Benzetilmiş Tavlama

Benzetilmiş Tavlama (Simulated Annealing), metallerin soğutularak minimum enerjili kristal yapıya dönüşmesi ile bir problemde minimumu arama işlemi arasında benzerlik kurarak optimizasyon problemlerine çözüm arar. Benzetilmiş Tavlama, Metropolis ve arkadaşlarının (1953) belirli bir ısı seviyesinde atomların dengeli dağılımını bulma amacıyla geliştirdikleri ve enerji değişimlerini taklit eden bir çalışmayı temel almaktadır. Benzetilmiş Tavlamanın bir optimizasyon tekniği olarak kullanılabilmesi fikri ise Kirkpatrick ve arkadaşları (1983) tarafından ortaya atılmıştır. Benzetilmiş Tavlama yaklaşımı, uygulama şeklinin basit olması ve geniş bir uygulama alanının bulunması sebebiyle birçok araştırmacı tarafından ilgi görmüştür. Benzetilmiş Tavlama metodunun temelinde yatan tavlama işlemi, katı bir maddenin gücünü artırmak için erime noktasını aşıncaya kadar ısıtılıp ardından soğutulması işlemi olarak tanımlanabilir. Soğuma hızına bağlı olarak katı maddenin yapısal özellikleri de değişiklik gösterir. Benzetilmiş Tavlamanın diğer yöntemlerden en önemli farkı, yerel minimum noktasına takılmama yeteneğidir. Bu yetenek Benzetilmiş Tavlamanın amaç fonksiyonundaki kötüye gitme durumunu da kabul edebilmesinden gelmektedir. Benzetilmiş Tavlama ise kötüye giden çözümleri de araştırabildiğinden yerel en iyiden çıkarak global en iyiye ulaşabilmektedir.

1.1.5. Tabu Arama Algoritmasi

Tabu Arama Algoritması, Glover (1986) tarafından ortaya atılan bir arama algoritmasıdır. Temel olarak Tabu Arama Algoritması; son çözüme götüren adımın, doğrusal hareketleri engellemek için bir sonraki iterasyonda tekrar kullanımının yasaklanması veya cezalandırılması şeklinde çalışmaktadır. Tabu arama, daha önce

incelenmiş çözümlerin oluşturduğu yol üzerindekiler hariç her çözümü inceleyebilen bir tekniktir. Bu sayede yeni çözümler incelenerek yerel minimum noktasından kaçınılmış olur. Algoritma temel olarak önce yerel minimum noktasına doğru hareket eder. Önceki hareketlerin tekrarlanmaması için bir veya birden fazla tabu listesi oluşturarak bu listeyi sürekli kontrol eder. Tabu listesinin asıl amacı önceden yapılmış hareketlerin tekrarı değil, ters yönde hareket yapılarak başa dönmeyi önlemektir (Cura, 2008a).

1.2. TEZİN AMAÇ VE KAPSAMI

Bu tezde; George B. Dantzig'in 1947 yılında simpleks metodunu geliştirmesinden bu yana kesin yöntemlerle çözülen doğrusal programlama problemleri meta sezgisel yöntemler kullanılarak çözdürülmeye çalışılmıştır. Karmaşık problemleri çözmede kullanılan meta sezgisel yöntemlerden olan GA, PSO, ACO, SA, TS optimizasyon algoritmalarının karşılaştırılarak sonuçların değerlendirilmesi hedeflenmektedir.

Optimizasyon problemlerinde karşılaşılan sorunlardan bir tanesi de hangi problem türü için hangi algoritmanın daha iyi sonuç verebileceğinin kestirilememesidir. Optimizasyon problem çözümünde doğru algoritmanın seçilmesi sonuç verileri için son derece büyük önem taşımaktadır. Bu bağlamda araştırmacılara, doğrusal programlama problemlerinin çözümü için optimizasyon algoritması seçiminde fikir sunmak ve zaman kaybını minimuma indirmek amaçlanmıştır.

Bu tezde bahsi geçen meta sezgisel yöntemler Branin ve Sphere fonksiyonları kullanılarak test edilmiş ve doğrulukları ispatlanmıştır. Rastgele oluşturulan 20 problem her bir algoritma için 30 kez çözdürülerek sonuç verileri alınmış, veri analizi için geliştirilmiş olan programa aktarılmış, grafikler ve çizelgeler halinde sonuç verileri incelenmiştir. Sonuç verileri incelenirken bağıl hata, ortalamaya en yakın değeri üreten ve minimum sonuç üreten çözümler ayrı ayrı çizelgeler halinde sunulmuştur.

1.3. TEZİN ORGANİZASYONU

Tez 6 bölümden oluşmaktadır. İkinci bölümde, optimizasyon ve doğrusal programlama kavramları anlatıldı. Doğrusal programlama modelinden bahsedilerek doğrusal

programlama için geçerli olan varsayımlar açıklandı. Doğrusal programlama problemlerinin çözümünde kullanılan tanımlamalar ve çözüm yöntemlerinden bahsedildi.

Üçüncü bölümde, meta sezgisel yöntemler genel olarak ele alındı ve tezde kullanılan GA, PSO, ACO, SA, TS optimizasyon algoritmaları tanıtıldı. Optimizasyon algoritmalarının özellikleri, çözüme ulaşmak için kullandıkları yöntem ve algoritma yapıları anlatıldı.

Dördüncü bölümde, doğrusal programlama problemlerinin çözümünde kullanılan meta sezgisel algoritmaların doğruluğu test edildi. Test işlemi için kullanılan Branin ve Sphere fonksiyonları tanıtıldı. Karşılaştırma için kullanılan doğrusal programlama problemlerinin tanımlamaları yapıldı ve doğrusal programlama problemleri için meta sezgisel yöntemler düzenlendi. Son olarak alınan verilerin analizi için hazırlanan program tanıtıldı.

Beşinci bölümde, tezde kullanılan meta sezgisellerin problem çözümünden elde ettikleri sonuç veriler çizelgeler halinde sergilendi. Karşılaştırma için bağıl hataların, ortalama sonuca en yakın sonucu elde eden çözüm değerinin, bu çözüme ulaşmak için harcanan sürelerin, minimum değere sahip olan çözümlerin ve bu çözüme ulaşmak için harcanan sürelerin verildiği çizelgeler listelendi.

Altıncı bölümde ise tezde yapılan çalışmalar değerlendirilerek gelecek çalışmalara ışık tutması için öneriler sunuldu.

2. GENEL KISIMLAR

İnsanoğlu, yaşamı boyunca eldeki kaynakları en verimli şekilde kullanmayı hedefler. "Bir ürün minimum maliyetle nasıl üretilir?" veya "Bir işletme maksimum karı nasıl yapar?" soruları, aslında çözülmesi gereken birer optimizasyon problemidir.

2.1. OPTİMİZASYON

Optimizasyonun çok çeşitli tanımı vardır. Matematiksel olarak optimizasyon; bir amaç fonksiyonunu, verilen tanım aralığında optimum(maksimum veya minimum) yapan değeri bulmaktır.

Yöneylem araştırmaları veya işletme bakımından optimizasyon, bir sistemi minimum maliyetle maksimum verimi elde edebilmek için düzenlemek olarak tanımlanabilir.

Optimizasyon, modelleme ve çözümleme olarak iki önemli bileşenden oluşur. Modelleme, gerçek yaşamda karşılaşılan bir sorunun matematiksel olarak ifade edilmesi; çözümleme ise çeşitli araçlar kullanılarak bahsi geçen modelin en iyi çözümünün elde edilmesidir (Afacan, 2007).

Bir optimizasyon problemi optimize edilecek bir amaç fonksiyonu veya maliyet fonksiyonu, değişkenlerin değer aralıkları ve kısıtlardan oluşur. Optimizasyon problemi tek bir dönem için çözülecekse statik, çözümde birden fazla dönem göz önünde bulundurulacaksa dinamik model olarak adlandırılır. Eğer optimizasyon problemindeki tüm karar değişkenleri pozitif reel değerlerden oluşuyorsa sürekli, tamsayı değerlerden oluşuyorsa kesikli optimizasyon problemleridir. Eğer karar değişkenlerinin kombinatoryal seçenekleri söz konusuysa kombinatoryal optimizasyon problemleri olarak adlandırılırlar. Eğer amaç fonksiyonu ve kısıtlar doğrusal olursa problem doğrusal programlama problemi olarak adlandırılır. Amaç fonksiyonu ile birlikte kısıtlardan bazıları ya da tümü doğrusal olmayan ifadelerden oluşuyorsa probleme doğrusal olmayan programlama problemi denilir.

2.2. DOĞRUSAL PROGRAMLAMA (LINEAR PROGRAMMING)

Doğrusal programlama, belirli doğrusal eşitlikler ya da eşitsizlikler şeklinde tanımlanan kısıtlayıcı koşullar altında doğrusal bir amaç fonksiyonunu maksimize ya da minimize etmek şeklinde tanımlanabilir (Bazaraa ve Jarvis, 1977).

II. Dünya Savaşı sırasında ve sonrasında, çeşitli projelerde ve kıt kaynakların etkin kullanımında planlama ve koordinasyonun zaruri olduğu anlaşılmıştır. ABD Hava Kuvvetleri ekibi SCOOP (Optimum Programların Bilimsel Hesaplanışı) tarafından yoğunlaştırılmış çalışma, Haziran 1947'de başlamıştır. Bunun bir sonucu olarak, 1947 yılı yazının sonunda George B. Dantzig tarafından simpleks metodu geliştirilmiştir. Doğrusal programlamaya olan ilgi; ekonomistler, matematikçiler, istatistikçiler ve devlet kurumları arasında çabucak yayılmıştır. 1949 yılı yazında Cowles Ekonomi Araştırma Komisyonu sponsorluğunda doğrusal programlama üzerine bir konferans yapılmıştır. Bu konferansta sunulan çalışmalar daha sonra 1951 yılında T. C. Koopmans tarafından "Üretim ve Dağıtım Faaliyet Analizi" isimli kitapta bir araya getirilmiştir (Bazaraa ve Jarvis, 1977).

Simpleks metodunun geliştirilmesinden bu yana pek çok kişi, matematiksel teorisini geliştirerek, etkin hesaplama metotları ve kodları geliştirerek, yeni uygulamalar bularak; diskrit, doğrusal olmayan, kombinatoryal, stokastik programlama problemleri ve optimal kontrol problemleri gibi daha kompleks problemleri çözmede doğrusal programlamayı yardımcı bir araç olarak kullanmış ve doğrusal programlamanın büyümesine katkıda bulunmuştur. Doğrusal programlamanın popülerliği; simpleks metodu ve bilgisayar yardımıyla kullanıcılarına büyük problemleri çok kısa sürelerde çözme imkânı sağlamasına, büyük ve karmaşık problemlerdeki yeteneğine ve bunun gibi birçok faktöre bağlanabilir (Bazaraa ve Jarvis, 1977).

Optimize etmek, belirli bir amaca en küçük masrafla ulaşmak ya da belirli kaynaklarla en fazla ürünü sağlamak anlamına gelir (Dantzig, 1963, Altay, 1990).

Doğrusal programlamadaki(LP) doğrusal (lineer) sözcüğü, modeldeki tüm matematiksel fonksiyonların doğrusal (lineer) olması gerektiğini belirtir. Programlama kelimesi ise

bilgisayar programlamaya işaret etmez; daha çok planlama ile eş anlamlıdır. Dolayısıyla doğrusal programlama, birçok uygun alternatif arasından belirlenmiş bir hedefe uyan optimal çözümü bulacak aktivitelerin planlanmasını içerir (Özçiloğlu, 2009).

2.2.1. Doğrusal Programlama Modeli

Genel olarak doğrusal programlama problemleri:

$$\text{Optimal } Z_{max/min} = \sum_{J=1} C_j x_j \sum_{J=1}^{n} C_j x_j \ , \quad X=(x_1, x_2, \dots, x_n) \tag{2.1}$$

Kısıtlar $a_{i1}x_1+a_{i2}x_2+\cdots+a_{in}x_n\ (\leq=\geq)b_i$, i=1,2,…,m

Doğal Kısıtlar $x_{j\geq 0}$, j=1,2,..,n

Şeklinde tanımlanır (Bazaraa ve Jarvis, 1977).

Burada

$Z_{max/min}$ amaç fonksiyonunu (maximizasyon ya da minimizasyon),

x_j karar değişkenlerini,

C_j sabit katsayılarını (parametreler),

a_{ij} i. kısıt için a değişkeninin, j.değeri katsayısını,

b_i i. kısıt için b değişkeninin değerini (sağ taraf sabitleri)

ifade eder.

2.2.2. Doğrusal Programlama Varsayımları

Bir doğrusal programlama probleminin içermek zorunda olduğu varsayımların bazıları aşağıda verilmektedir.

2.2.2.1. Doğrusallık Varsayımı

Doğrusal programlama modelinde, fonksiyona giren ve çıkan madde miktarları arasında doğrusal bir ilişki vardır. Eğer çıkan madde miktarını ikiye katlamak istersek fonksiyona giren tüm ilgili madde miktarlarını ikiye katlamamız yeterli olacaktır (Dantzig, 1963).

2.2.2.2. Toplanabilirlik Varsayımı

Değişik kısıtlara kaynak olan girdilerin toplamının her bir işlem için ayrı ayrı kullanılan girdilerin toplamına eşit olduğunu gösterir.

2.2.2.3. Negatif Olmama Varsayımı

Doğrusal programlamada yer alan temel, aylak ve artık değişkenlerin değeri sıfır ya da sıfırdan büyük olmalıdır (Özçiloğlu, 2009).

2.2.2.4. Sınırlılık Varsayımı

Üretimde kullanılan kaynaklar sınırlıdır. Bu nedenle üretim miktarı da kaynaklardan dolayı sınırlı olacaktır.

2.2.3. Doğrusal Programlama Problemlerinin Çözümünde Kullanılan Tanımlamalar

2.2.3.1. Konveks Set

Bir nokta kümesini gösteren S içindeki bütün P_1,P_2 nokta çiftlerini birleştiren doğru parçaları S kümesi içinde yer alıyorsa bu küme konveks settir.

2.2.3.2. Uygun Çözüm

Doğrusal bir karar modelinin tüm negatif olmama ve sınırlayıcı şartlarını sağlayan her x vektörüne, uygun çözüm denilmektedir. Bu uygun çözümlerin oluşturduğu kümeye ise uygun çözüm alanı denilir.

2.2.3.3. Temel Uygun Çözüm

Değişken sayısı n, kısıt sayısı m olan bir doğrusal programlama probleminde; n>m olmak üzere, n-m kadar değişken sıfır olarak alınırsa m bilinmeyenli m tane denklem elde edilmiş olur. Bu denklem sisteminin katsayılar determinantı sıfırdan farklı olmak kaydıyla elden edilen çözüme temel çözüm denir. Temel çözümler arasında negatif olmama şartını sağlayanlara ise temel uygun çözüm denir.

2.2.3.4. Optimum Çözüm

Doğrusal programlamada, amaç fonksiyonunu maksimize ya da minimize eden çözüme optimum çözüm denir.

2.2.4. Doğrusal Programlama Problemlerinin Çözüm Yöntemleri

Doğrusal programlama problemlerinin çözümünde kullanılan çeşitli yöntemler vardır. Bu yöntemlerin en bilinenleri grafik yöntem ve simpleks yöntemidir.

Grafik yöntemde iki temel adım kullanılır. İlk adım, modelin tüm kısıtlarının sağlandığı uygun çözümleri içeren çözüm uzayının belirlenmesi; ikinci adım ise belirlenen çözüm uzayındaki tüm noktalar arasından optimum çözümün belirlenmesidir.

Bir doğrusal programlama probleminin optimum çözümü, çözüm uzayının köşe noktaları arasındadır. Bu köşe noktası matematiksel olarak en uç nokta olarak bilinir. Grafik yöntem bu durumu açıkça göz önüne sermektedir. Aynı zamanda bu sonuç, herhangi bir doğrusal programlama problemini çözmek için geliştirilmiş genel bir matematiksel yöntem olan simpleks yönteminin de temel kavramıdır.

Simpleks yönteminde matematiksel olarak en uç noktayı tarif edebilmek için öncelikle modelin standart doğrusal programlama problemi haline getirilmesi gerekir. Bunun için modelde eşitsizlik halinde bulunan kısıtların dolgu ya da artık maddeler kullanılarak eşitliklere dönüştürülmesi sağlanır. Model standart doğrusal programlama problemi şeklini aldıktan sonra ise doğrusal eşitlikler haline gelmiş kısıtların eşanlı temel çözümlerine bakılır. Bu temel çözümler, çözüm uzayındaki en uç noktaların hepsini tam olarak tanımlar. Son olarak da simpleks algoritması, bu temel çözümler içinden optimum olanının yerini belirler.

Doğrusal programlama problemlerinin çözümü için; simpleks yöntemi, üst ve alt sınırlandırma, ayrıştırılmış ve parametrik programlama gibi etkin hesaplama yöntemlerinin yanı sıra bu yöntemlerden tamamen farklı olan Karmarkar'ın iç nokta algoritması da kullanılabilmektedir.

Bazı matematiksel modeller, bilinen herhangi bir optimizasyon algoritması ile çözülemeyecek derecede karmaşık olabilmektedir. Bu nedenle optimum çözümü aramak son derece zahmetli olmaktadır. Bu tür durumlarda optimum çözümü aramak yerine sezgisel yöntemler kullanılarak iyi bir çözümü arama yoluna gidilebilir (Taha, 2007).

3. METASEZGİSEL YÖNTEMLER

Meta sezgisel (Meta Heuristic) kelimesi ilk defa Glover (1986) tarafından kullanılmıştır. Yunanca sonra anlamına gelen "meta" ve sezgisel anlamına gelen "heuristic" kelimelerinin birleşiminden oluşmaktadır. Stützle (1997) meta sezgisel yöntemleri şöyle tanımlamıştır: "*Meta sezgiseller, probleme özgü sezgisellere performanslarını arttırmak için rehberlik eden üst düzey stratejilerdir. İterasyonlarla ilerleyen yerel aramanın yerel bir optimum noktada saplanıp kalmasına engel olmak temel amaçlarıdır. Bunun için aramanın daha kötü çözümlere geçiş yapmasına izin verebilir veya yerel arama için oluşturulan başlangıç çözümlerinin tamamen rassal olarak değil de daha zeki bir yöntemle oluşturulmasını sağlayabilirler. Çoğu meta sezgisel yöntem, yüksek kalitede çözümleri çabukça bulabilen eğilimleri öneren yöntemler olarak yorumlanabilir. Önerilen eğilim değişik formlarda olabilir. Amaç fonksiyon değerini temel alan azaltma eğilimi, daha önceki kararları temel alan bellek eğilimi veya geçmiş performanslara dayalı deneyim eğilimi gibi örnekler verilebilir. Meta sezgisellerin çoğu arama esnasında olasılıklara dayanan seçimler yaparlar. Ancak şansın önemli olduğu rassal arama süreçlerinden farklı olarak seçim olasılıkları zeki bir seçim yapılabilmesini sağlayacak şekilde oluşturulur.*" (Kılıç, 2008).

Meta sezgisel yöntemler zor ve karmaşık optimizasyon problemlerde kaliteli çözümlere ulaşabilmek için sezgisel yaklaşımları kullanan tasarımlar olarak da adlandırılabilir (Kılıç, 2008). Çözüm uzayının büyüklüğü ya da optimizasyon modelindeki değişken ve kısıt sayılarının fazlalığı nedeniyle kesin sonuç alınan yöntemlerin kullanılamadığı durumlarda, uygulamaya elverişli olmayan problemlerin çözümünde, meta sezgisellerin kullanılması giderek yaygınlaşmaktadır. Kesin sonuçların alındığı yöntemlerden farklı olarak meta sezgiseller, optimum yerine optimuma yakın çözümler elde edilmesini sağlar (Aydın, 2009).

Klasik sezgisel yöntemlerle karşılaştırıldığında meta sezgisel yöntemlerin en büyük avantajı ise probleme özgü tasarlanmamış olmaları, temel adımları genel olarak

tanımlanmış olması ve problemlerin özelliklerinden bağımsız olarak kullanılabilmeleridir. Bu özellikleri sayesinde meta sezgisel yöntemler hemen her türlü probleme uyarlanabilmektedirler. Bu durumla birlikte, bir problem için özel tasarlanmış bir metot kadar iyi sonuç alınamayacağı ihtimali ise bir dezavantaj oluşturmaktadır (Aydın, 2009).

Meta sezgisel algoritmaların önemli bir grubunu popülasyon temelli algoritmalar oluşturmaktadır. Bu gruptaki algoritmalar, probleme ait aday çözümleri barındıran bir popülasyon oluştururlar ve bu popülasyonu çeşitli operatörler yardımıyla geliştirerek yaklaşık çözümler üretirler. Popülasyon temelli algoritmaları, gelişime dayalı ve sürü zekâ temelli algoritmalar olarak iki alt gruba bölmek mümkündür. Genetik Algoritma (GA), Evrim Stratejileri (ES), Evrimsel Programlama (EP), Genetik Programlama (GP) gelişime dayalı algoritmalara; Karınca koloni algoritması (ACO), Isıl İşlem (SA), Tabu Araştırma (TS), Parçacık Sürü Optimizasyon algoritması (PSO) sürü zeka temelli algoritmalara örnek olarak gösterilebilir (Akay, 2009).

3.1. GENETİK ALGORİTMA

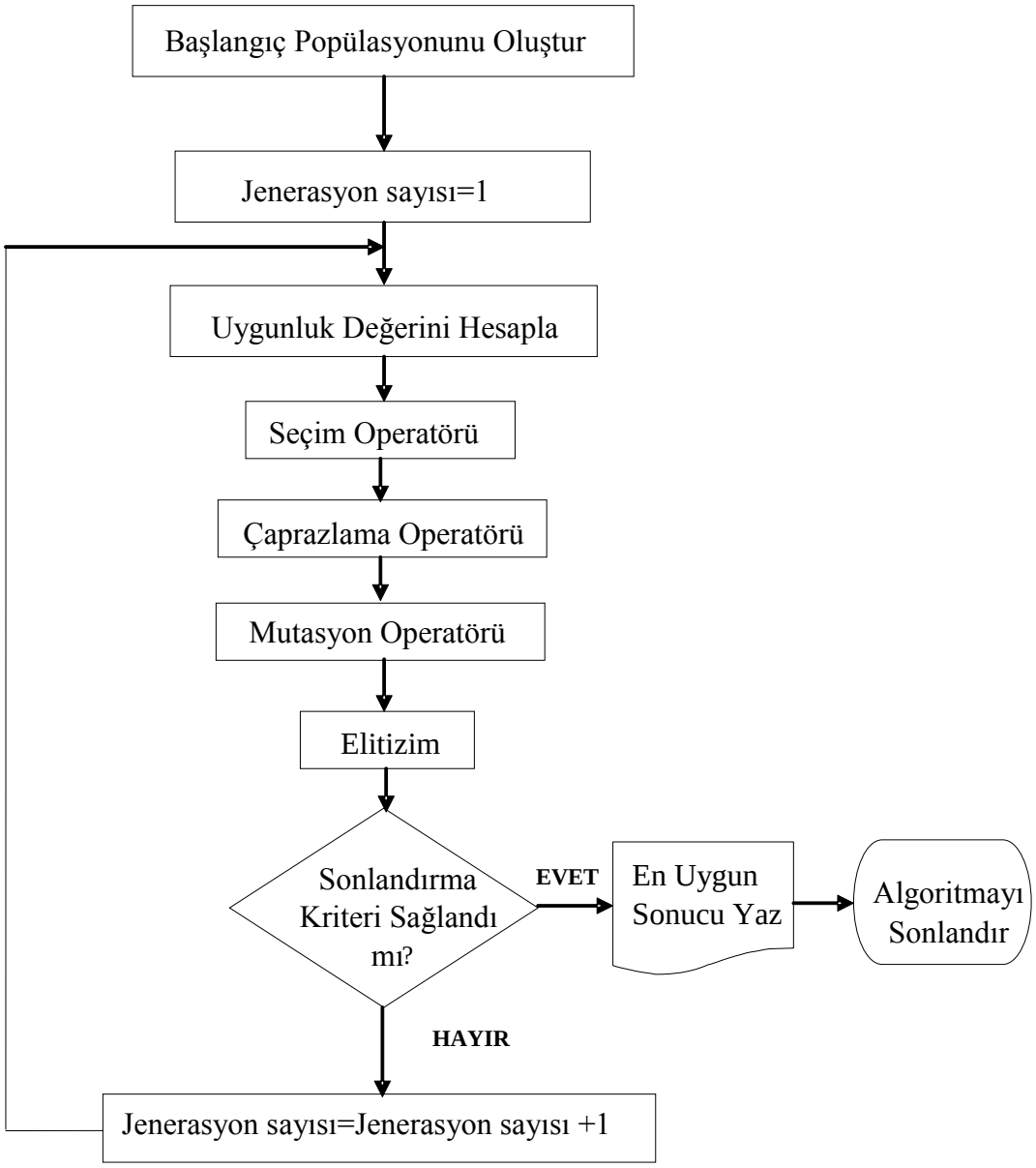
1950 ve 1960'larda çok sayıda bilgisayar mühendisi, "Evrim, mühendislik problemleri için bir optimizasyon aracı olarak kullanılabilir." fikrinden yola çıkarak evrimsel sistemler üzerine çalıştı. Bütün bu sistemlerin ana fikri belirli bir soruna, doğal genetik varyasyon ve doğal seleksiyondan esinlenen operatörleri kullanarak aday çözümlerden bir nüfus geliştirmekti (Mitchell, 1999).

Genetik Algoritmalar 1960'lı yıllarda John Holland tarafından bulunmuş ve 1960-1970'li yıllarda Michigan Üniversitesinde Holland, öğrencileri ve meslektaşları tarafından geliştirilmiştir. Holland'ın asıl amacı, belirli problemleri çözmek için algoritma tasarlamak değil; adaptasyon olgusu üzerinde çalışmak ve doğal adaptasyonun bilgisayar sistemlerinde kullanılabilmesini geliştirmek olmuştur.

Holland'ın GA metodu, bir popülasyondaki kromozomları bir tür doğal seleksiyon kullanarak çaprazlama (crossover) ve mutasyon gibi genetik operatörler ile yeni popülasyona almak için geliştirilmişti. Seçme operatörü ile o popülasyondaki iyi

bireyler seçilmekte ve bu iyi bireyler kendi aralarında çaprazlanarak yeni bir birey oluşmaktaydı. Oluşturulan bu birey sonraki popülasyona aktarılmakta idi. John Holland; mutasyon, çaprazlama gibi kavramları literatüre tanıştıran kişi olması bakımından önemlidir. Holland, 1975 yılında çıkardığı "Adaptation in Natural and Artificial Systems" adlı kitabında genetik algoritmalar altında adaptasyon için teorik bir çerçeve sunmuştur (Mitchell, 1999, Gülcü, 2006).

3.1.1. Genetik Algoritmanın Çalışması



Şekil 3.1: GA akış diyagramı

GA'nın genel anlamda akış diyagramı Şekil 3.1'de gösterilmektedir. İlk olarak uygunluk fonksiyonu, ardından rastgele değerlerden oluşan başlangıç popülasyonu oluşturulur. Popülasyondaki tüm bireylerin uygunluk değerleri hesaplanır. Uygunluk değerleri hesaplanan bireyler sırasıyla seçim, çaprazlama, mutasyon ve elitizm operatörleriyle işleme tabi tutulur. Bütün bu işlemler sonucunda eğer sonlandırma kriteri sağlandıysa en uygun sonuç ekrana yazdırılarak algoritma sonlandırılır. Eğer sonlandırma kriteri sağlanmadıysa jenerasyon sayısı bir artırılarak kriter sağlanıncaya kadar işlem devam ettirilir.

3.1.2. Başlangıç Popülasyonunun Oluşturulması

Genetik algoritma uygulamalarında ilk adım başlangıç popülasyonunun belirlenmesidir. Bahsi geçen popülasyon için; her bir değişken geni oluştururken, genlerin birleşimi kromozomu ve kromozomların birleşimi ise popülasyonu oluşturmaktadır. Başlangıç popülasyonu oluşturulurken işe popülasyondaki birey sayısını belirleyerek başlanmaktadır. Birey sayısı için bir standart yoktur. Birey sayısının fazlalığı, yapılan işlemlerin karmaşıklığı ve aramanın derinliği ile ilgilidir.

Birey sayısı belirlendikten sonra popülasyon oluşturma işlemine geçilir. Başlangıç popülasyonu genellikle rastgele bireylerden oluşturulur. Ancak problemle ilgili olarak bazı çözümler tahmin ediliyorsa başlangıç popülasyonu bu çözümler doğrultusunda da oluşturulabilir (Coşkun, 2006).

3.1.3. Uygunluk Değerinin Hesaplanması

Genetik algoritmada iyi bireylerin özelliklerinin nesilden nesle aktarılması için uygunluk değerine [f(x)] ihtiyaç vardır (Tozan, 2007). Uygunluk değerlerinin hesaplanması popülasyonun her bir bireyi için ve popülasyon sayısı kadar ayrı ayrı gerçekleştirilir. Uygunluk değerleri hesaplandıktan sonra içlerinden başarılı olan popülasyon adımları seçilerek başarılı bireyler arasından yeni bir nesil oluşturulur. Yeni neslin oluşturulmasında evrim teorisindeki seçim, çaprazlama, mutasyon gibi etkiler kullanılır. Bahsi geçen işlemlerden sonra yeni bir popülasyon oluşturulur ve uygunluk değerleri bu popülasyon için tekrar hesaplanır. İstenilen sonuca ulaşıncaya kadar bu döngü tekrar eder (Öztürk, 2007).

3.1.4. Kodlama Yöntemleri

Genetik algoritmada kromozomların yeterli bilgi içerecek şekilde, birer dizi halinde kodlanması gerekir. Bu kodlama şekli genetik algoritmanın başarısını etkileyen faktörlerdendir (Gülcü, 2006). En yaygın kullanılan kodlama yöntemi ikili kodlamadır (Coşkun, 2006).

Kodlama yöntemlerinde ikili kodlama gibi, çok karakterli ve nümerik değerli kodlama çeşitleri de bulunmaktadır. Seçilecek olan kodlama yöntemi, problemi en basit şekilde ifade etmelidir (Gülcü, 2006).

3.1.5. Elit Bireylerin Seçilmesi

Genetik algoritmada en iyi uygunluk değerine sahip popülasyon adımı ya da adımlarının bir sonraki popülasyona aktarılması için elitizm operatörü bu değerleri seçer ve bir sonraki jenerasyonda kullanmak üzere kaydeder (Wook ve diğ., 2003). Böylelikle en iyi uygunluk değerine sahip olan birey ya da bireylerin bir sonraki jenerasyonda da yaşamını sürdürmeleri garanti edilmiş olur (LU, 2003). Jenerasyondaki diğer bireyler ise genetik algoritmanın diğer operatörleri olan seçim, çaprazlama ve mutasyon yolu ile belirlenir (Man ve diğ., 1996, Elkamchouchi ve Wagib, 2001, Öztürk, 2007).

3.1.6. Seçim Operatörleri

Uygunluk fonksiyonuna göre yeni nesli oluşturacak bireylerin seçilmesi işlemine genetik algoritmanın seçim operatörü denir. Seçim işleminin amacı başarılı bireylerin yaşamlarını sürdürebilmelerini ve yeni neslin bu bireylerden oluşmasına imkân vermek, başarısız olan bireylerin ise elenmesini sağlamaktır. Seçim işlemi için; rulet çarkı seçimi, turnuva seçimi ve sıralama seçimi gibi birçok yöntem geliştirilmiştir (Öztürk, 2007).

3.1.6.1. Rulet Çarkı Metodu

Rulet çarkı metodunda bireyler uygunluk fonksiyonu değerlerine göre bir çarkın dilimlerini oluştururlar (Öztürk, 2007). Bu işlem için tüm bireylerin uygunluk değerleri bir tabloya kaydedilir ve tüm değerler toplanır. Her bireyin uygunluk değeri toplam uygunluk değerine bölünerek yüzdelik dilimleri hesaplanır (Bolat, 2006). Uygunluk

değeri yüksek olan birey, çark üzerinde daha geniş bir alana sahip olur ve seçilme ihtimalleri artar; fakat gene de seçilme garantileri yoktur (Küçüktezcan, 2008, Kaya, 2006). Elitizm kullanılmıyorsa çark birey sayısı kadar, kullanılıyorsa toplam birey sayısının elitizmde belirlenen birey sayısı eksiği kadar çark çevrilir (Mastorakis ve diğ., 2003).

3.1.6.2. Turnuva Seçim Metodu

Turnuva seçim metodunda, popülasyondan rastgele seçilen bireyler gruplara katılır. Bir gruba seçilmeyen bireyler başka bir gruba seçilme olanağı bulacağından tüm bireylerin seçilme şansları olur (Srinivas ve Deb, 1994, Back ve diğ., 1997). Grup içerisinde en iyi uygunluk değerine sahip olan birey ebeveyn olarak seçilir. Bu işlem seçilen bireyler gerekli sayıya ulaşılıncaya kadar devam eder. Ebeveyn olarak belirlenen bireyler çaprazlama ve mutasyon işlemleriyle yeni jenerasyonu meydana getirirler (Öztürk, 2007).

3.1.6.3. Sıralı Seçim Metodu

Sıralı seçim metodunda bireyler, uygunluk değeri en kötüden en iyi olana doğru sıralanırlar. Bu sıralama doğrultusunda eşleşmeler gerçekleştirilir. Zayıf bireylerin de seçilme şansı bulması bu yöntemin en büyük avantajıdır. Bu sayede güçlü bireylerin nüfus üzerinde baskın olmaları da engellenmiş olur (Öztürk, 2007) .

3.1.7. Çaprazlama Operatörleri

Genetik algoritmanın en önemli operatörü çaprazlama operatörüdür. Çaprazlama işleminden, bireylerdeki iyi özellikleri birleştirip daha iyi çözümlere ulaşması beklenir.

Genel olarak 4 çeşit çaprazlama operatörü kullanılır.

Tek nokta çaprazlama: Genetik algoritmada kullanılan en basit çaprazlama türüdür. Rastgele seçilen bir noktadan sonraki genler yer değiştirilerek yeni bireyler oluşturulur (Park ve diğ., 2000).

İki nokta çaprazlama: Rastgele seçilen iki nokta arasındaki genler yer değiştirilerek yeni bireyler oluşturulur.

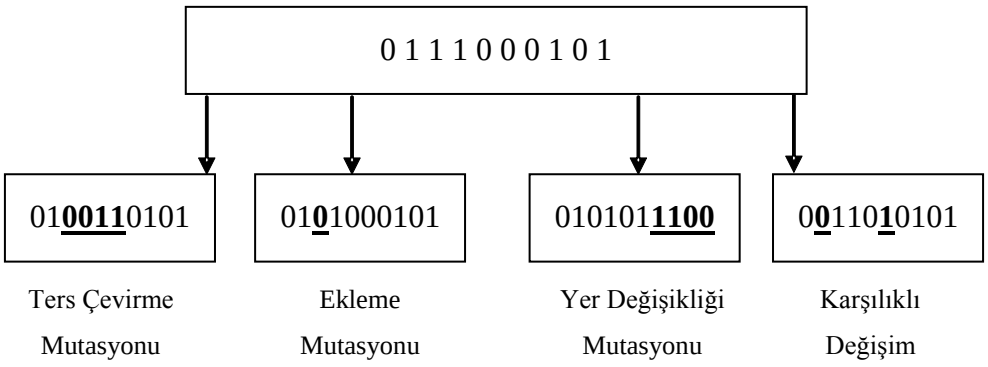
Çok nokta çaprazlama: İki nokta çaprazlama metodunun, aynı işlem üzerinde birden fazla defa kullanılmasıyla yeni bireyler oluşturulur (Bolat, 2006).

Düzenli çaprazlama: Bireyleri gen sayısına eşit, rastgele değerlerden oluşan bir çaprazlama maskesi oluşturulur. Çaprazlama maskesinde "1" değeri yeni oluşturulacak 1. bireyin genini 1. ebeveynden, 2. bireyin genini 2. ebeveynden; "0" değeri ise 1. bireyin genini 2. ebeveynden, 2. bireyin genini 1. ebeveynden kopyalayacağı anlamına gelir (Bolat ve diğ., 2004).

3.1.8. Mutasyon Operatörü

Genetik algoritmada işlemler esnasında bireylerin gen diziliminin birbirine çok yaklaşması çeşitliliğin azalması anlamına gelir. Çeşitliliğin azalması, üretilen çocukların ebeveynlerine benzemeleri demektir ve yeni bireylerin üretimi durabilir (Öztürk, 2007). Bahsi geçen durumu önlemek için mutasyon operatörü kullanılır ve çeşitlilik sağlanmış olur.

Çözüm aranan problemin yapısına bağlı olarak ters çevirme, ekleme, yer değiştirme, karşılıklı değişim mutasyon operatörlerinden biri kullanılabilir.



Şekil 3.2: Mutasyon yöntemleri ve operatörleri (Bolat ve diğ., 2004).

Mutasyon operatörü uygulamaları Şekil 3.2'de görülmektedir. Şekil üzerinde altı çizili olarak verilen değerler, mutasyona uğramış genleri göstermektedir. Ters çevirme mutasyonunda, kromozom üzerinde rastgele iki nokta belirlenir ve bu iki nokta arasındaki genler tersten yazılır. Ekleme mutasyonunda, rastgele seçilen bir eleman yine

rastgele bir yere yerleştirilir. Yer değişikliği mutasyonunda, rastgele belirlenen iki nokta arasındaki genler; ekleme mutasyonunda olduğu gibi rastgele bir yere yerleştirilir. Karşılıklı değişim mutasyonunda ise rastgele seçilen iki genin yerleri değiştirilir (Bolat ve diğ., 2004).

3.2. KARINCA KOLONİSİ OPTİMİZASYONU

1989 yılında Goss ve arkadaşları (1989) laboratuvar ortamında yetiştirdikleri karınca kolonilerini izleyerek çalışmalar gerçekleştirmişlerdir. Bu çalışmalardan bazıları ise kolonilerin incelenerek onlardan esinlenen yeni yöntem ve algoritmaların bulunması üzerine yoğunlaşmaktadır (Erdoğan, 2008). Karınca koloni optimizasyonu ise ilk kez 1990'ların başında Dorigo ve meslektaşları tarafından zor kombinatoryal problemlerin çözümü için doğadan esinlenilmiş bir meta sezgisel olarak önerilmiştir (Dorigo ve Blumb, 2005).

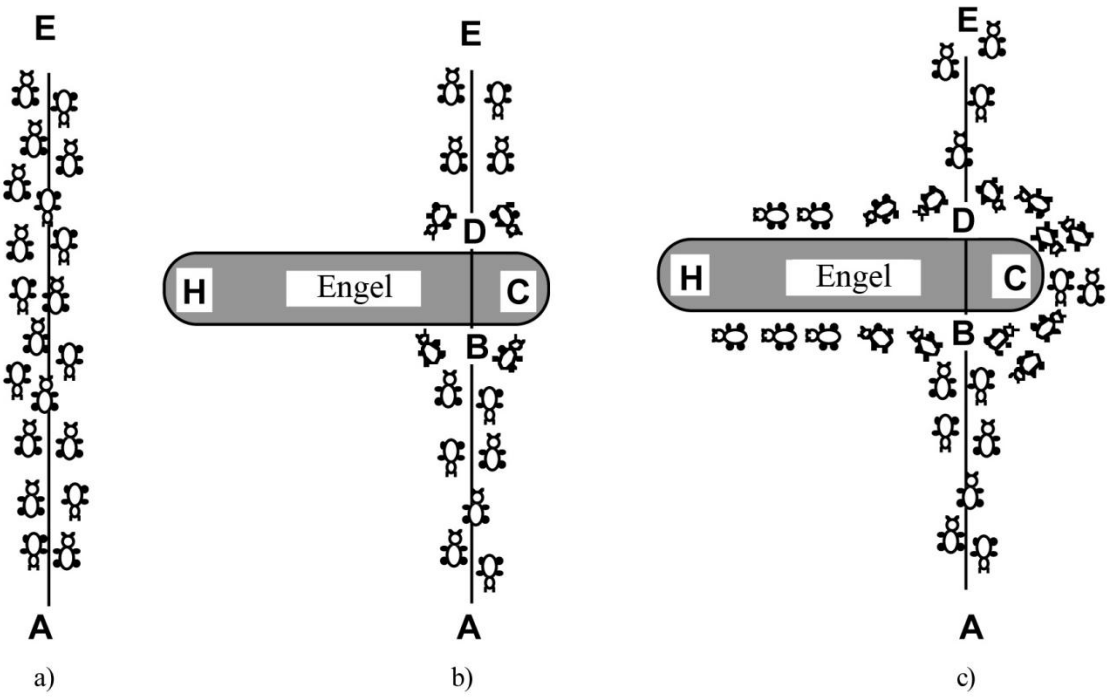
Karınca algoritmaları gerçek karınca kolonilerinden esinlenilerek oluşturulmuştur. Karıncalar sosyal böceklerdir. Koloniler halinde yaşayan böceklerde bireylerin davranışları bütünsel olarak kolonin hayatta kalmasına yöneliktir. Bireysel olarak incelendiğinde bu böceklerin hareketleri anlamsız olarak nitelendirilebilirken kolonin tamamı gözlendiğinde en iyiyi arama yönünde olduğu söylenebilir. Karınca kolonilerinin en önemli ve ilginç davranışı ise yiyecek arama davranışları, özellikle de onların yiyecek ve yuvaları arasındaki en kısa yolu bulmalarıdır (Dorigo ve diğ., 1999).

3.2.1. Gerçek Karıncalar

Karıncalar görme yetileri olmayan böceklerdir. Yiyecek ararken rastgele bir biçimde yuvalarının etrafını tararlar. Bir karınca yiyecek kaynağı bulduğunda ise yiyeceğin besin kalitesini ve miktarını değerlendirir, sonra da yiyeceğin bir kısmını yuvaya taşır. Dönüş yolunda karınca yere feromon isminde bir kimyasal iz bırakır. Karıncalar için feromon bir uyarandır ve karıncalar feromon miktarının fazla olduğu bölgeye yönelme eğilimi gösterirler. Karıncanın dönüş yolunda bıraktığı ve yiyecek kalitesi ile miktarını ifade eden feromonun miktarı diğer karıncalara yiyecek kaynağını bulmalarında rehberlik eder. Feromon salgılayarak ve ortamdaki feromon miktarını algılayarak karıncalar

arasında oluşturulan bu dolaylı iletişim şekli yiyecek ve yuvaları arasındaki en kısa yolu bulmalarını sağlar. Gerçek karınca kolonilerinin bu özelliği, yapay karınca kolonileri ile koloni optimizasyon problemlerini çözmek için kullanılır (Dorigo ve Blumb, 2005).

Karınca kolonilerine bütün olarak bakıldığında yiyecek arayan karıncaların bir engelle karşılaşma durumlarında bile feromonlar sayesinde haberleşerek en uygun yolu buldukları gözlenmektedir. Gerçek karıncaların yiyecek ve yuvaları arasındaki davranış şekilleri Şekil 3.3'de gösterilmiştir.



Şekil 3.3: Gerçek Karınca Davranışları (Dorigo ve diğ., 1996).

Şekil 3.3 üzerinde A noktaları yuvayı, E noktaları ise yiyeceği temsil etmektedir. a) Karıncaların doğrusal ve en kısa yolu kullanarak yuvaları ile yiyecek arasındaki hareketlerini göstermektedir. Bir karınca yiyeceği bulduğunda dönüş yolunda feromon salgılar. Diğer karıncalar ise bu feromonun kokusunu alarak aynı yolu takip ederler. Aynı yolu takip eden karınca sayısı arttıkça yol üzerindeki feromon miktarı da artacaktır. Bu durum gerçekte görmemelerine rağmen önlerine bir engel çıktığında bile gıdaya giden en kısa yolu bulmalarını açıklamaktadır. b) Karıncaların yuvaları ve yiyecekler arasına bir engel konulmuştur. Böyle bir durumla karşılaşan bir karınca

yoluna devam edemez. Karınca, B ve D noktaları arasında H ve C yollarından birini seçmek durumunda kalır. Böyle bir durumda H ve C yollarının seçilme olasılıkları eşittir. Karınca yaptığı seçime göre yoluna devam eder ve kendi yolunu tekrar çizer. c) Yanlış olan B-H-D yolunu seçmiş olan karınca, doğru olan B-C-D yolunu seçen karıncaya oranla yiyeceğe geç ulaşacaktır. Dönüşte, B-H-D yolunu seçmiş olan karınca D noktasına geldiğinde tekrar bir seçim yapmak zorunda kalacaktır. Fakat B-C-D yolunu seçmiş olan karınca dönüş yolunu çoktan geçmiş olduğu için o yol üzerindeki feromon miktarı B-H-D yoluna oranla daha fazla olacaktır. Dolayısıyla yiyeceğe giderken B-H-D yolunu seçmiş olan karıncanın dönüş yolunda B-C-D yolunu seçme olasılığı daha fazladır. Aynı yolu kullanan karınca sayısı arttıkça feromon miktarı da artacağından kısa olan B-C-D yolu kendiliğinden belirlenmiş ve çok daha fazla kullanılmaya başlanmış olur.

3.2.2. Yapay Karıncalar

Yapay karınca koloni optimizasyonu gerçek karıncalar model alınarak gerçekleştirilmiştir. Gerçek karıncaların bazı özellikleri aynen alınmıştır:

- Birbiriyle, feromon maddesi aracılığıyla dolaylı olarak iletişim kuran bireylerin oluşturduğu koloni.
- Yerel haberleşme için feromon maddesinin kullanılması.
- En kısa yolu bulmak için gerçekleştirilen bir dizi yerel hareket.
- Yerel bilgiyi kullanan ve geleceği hesaplamayan değişken bir karar politikası.

Gerçek karıncalarda bulunmayan bazı özellikler ise yapay karınca kolonilerine eklenmiştir:

- Zamanın ayrık olarak hesaplandığı bir ortamda yaşarlar ve hareketleri ayrık durumdan ayrık duruma geçişleri içerir.
- Belli bir hafızaları vardır ve bireyler önceki hareketlerini bu hafızada tutabilirler.
- Oluşturdukları çözüm kalitesinin bir fonksiyonu miktarında feromon biriktirirler.

- Birçok durumda yapay karıncalar sadece bir sonuca ulaştıklarında feromon izlerini günceller.
- Sistemin genel etkinliğini artırabilmek için, karınca koloni optimizasyon algoritmaları ileri bakabilme, lokal optimizasyon, geriye dönük iz sürebilme gibi fazladan kabiliyetlerle zenginleştirilmiştir (Dorigo ve diğ., 1999).

3.2.3. Karınca Koloni Algoritmalarının Sınıflandırılması

3.2.3.1. Karınca Sistemi

Karınca Sistemi algoritması 1991 yılında Dorigo ve arkadaşları tarafından ortaya atılmış olan ilk karınca algoritmasıdır (Dorigo ve diğ., 1991). Karınca Sistemi algoritmasının en önemli özelliği kendinden sonra gelen karınca algoritmalarına prototip teşkil etmesidir. Karınca sisteminde bir karıncanın iki nokta arasında hareket olasılığı, uygun olmayan çözüme giden tüm hareketlerin engellenmesi için 0'a eşitlenir. Bunun için bir karıncanın bir noktadan diğer noktalara tüm uygunsuz hareketlerini içinde barındıran bir *yasak hareketler listesi* tutulur.

3.2.3.2. Max-Min Karınca Sistemi

Karınca sistemi algoritması Stützle ve Hoos tarafından geliştirilerek Max-Min Karınca Sistemi olarak ortaya çıkmıştır (Stützle ve Hoos, 1997). Karınca Koloni optimizasyon algoritmalarında yapılan araştırmalar, en iyi çözüme sahip olan karıncayı çözüm araştırması sırasında daha fazla kullanmanın performansı olumlu yönde artırdığını göstermektedir. Stützle ve Hoos Karınca Kolonileri optimizasyonunun en iyi performansına ulaşma yönteminin; arama sırasında bulunan en iyi sonuçların daha fazla kullanılması olabileceğini önermişlerdir.

3.2.3.3. Karınca Koloni Sistemi

Gambardella ve Dorigo (1995) tarafından geliştirilen Ant-Q algoritmasından sonra 1996 yılında Gambardella ve Dorigo algoritmayı daha da geliştirerek Karınca Koloni Sistemi algoritmasını ortaya atmışlardır (Gambardella ve Dorigo, 1996). Karınca Koloni Sistemi algoritmasının en önemli katkısı ise bir iterasyon tamamlandıktan sonra gerçekleşen feromon güncellemesine ek olarak yerel feromon güncellemesi kavramını ortaya atmış olmasıdır.

3.2.3.4. Popülasyon Temelli Karınca Koloni Optimizasyonu

Popülasyon Temelli Karınca Kolonisi (P-KKO) optimizasyonu ilk olarak Guntsch ve Middendorf (2002) tarafından ortaya atılmıştır. P-KKO'u yazılım geliştirme esnasında sıralı bir yapı sağlama ve dinamik optimizasyon problemlerin çözüm arama üzerinde yoğunlaşmaktadır. P-KKO feromon değerlerinin tamsayılardan oluşması ve belirli bir değer aralığında sıkıştırılmasını önermiştir. P-KKO mevcut popülasyona yeni birey eklemek için pozitif feromon güncelleme, mevcut popülasyondan birey çıkartmak için ise negatif feromon güncelleme yöntemini kullanmaktadır. P-KKO'nun en önemli özelliği ise sadece bir önceki iterasyondan çıkan en önemli bilginin diğer iterasyona aktarılmasıdır. Karınca Koloni Optimizasyonunda feromon matrisinin geçmiş iterasyonlarının tamamı dikkate alınırken, P-KKO'nda geçmiş iterasyonlarda elde edilen en iyi sonuçlar kümesinin oluşturduğu bir popülasyon dikkate alınır.

3.2.3.5. Sürekli Global Optimizasyon ve Karınca Kolonisi Optimizasyonu

Karınca Kolonisi Optimizasyon algoritmaları atama, programlama, grafik renklendirme, en kısa yol, gezgin satıcı gibi kesikli optimizasyon problemlerinde sıklıkla kullanılmasına karşın sürekli global optimizasyon problemlerinde daha az kullanılmaktadır (Toksari, 2006). Global minimum noktayı arayan bir Karınca Koloni Optimizasyon algoritması önermiştir (Cura, 2008b).

3.2.4. Algoritma

Algoritmanın ilk adımı olarak m adet karınca, m adet rastgele başlangıç vektörü ile konumlandırılır ($x^k_{initial}$, k=1,2,…,m). Ardından feromon izine dayalı değişkenler aktarılır. Bu algoritmada, fermon miktarı sadece önceki iterasyonlardan elde edilen en iyi objektif fonksiyon değerinin etrafında yoğunlaşır. Her iterasyonun öncesinde, her bir karıncanın çözüm vektörü Denklem (3.1) ile hesaplanır.

$$X_t^k = x_{t-1}^{best} \pm dx \quad (t = 1,2,\dots,I) \tag{3.1}$$

X_t^k çözüm vektörü t iterasyonundaki k numaralı karıncayı ifade eder. x_{t-1}^{best}, (*t-1*) numaralı iterasyonda belirlenmiş en iyi çözümü ve dx ise sıçrama uzaklığını belirten $[-\alpha, \alpha]$ aralığında rastgele üretilmiş bir vektördür.

Denklem (3.1)'de ; (+) işareti sadece x koordinat düzlemine göre X_t^k noktasının, global minimum noktasının solunda kaldığı durumlarda kullanılır. (-) işareti ise tam tersi X_t^k noktasının, global minimum noktasının sağında kaldığı durumlarda kullanılır.

İlerleme yönü Denklem (3.2) ile belirlenir.

$$\bar{x}_{initial}^{best} = x_{initial}^{best} + \left(x_{initial}^{best} \times 0.01\right) \quad (3.2)$$

Eğer $f\left(\bar{x}_{initial}^{best}\right) \leq f(x_{initial}^{best})$ ise (+) işareti, aksi durumda (-) işareti kullanılır. ($\pm$) işareti ilk çözüm sonrasında gözlemlenen hareketin yönünü belirler (Terzi ve Serin, 2010).

Initialization

FOR *i*=1 TO *I* (*I*=iterasyon sayısı)

IF *i*=1 THEN *m* tane rastgele karınca üret

ELSE $[-\alpha, \alpha]$ aralığında hareket vektörü belirle

END IF

FOR *i*=1 TO *m*

Belirle($f(x_{\mathrm{t}}^{best})$)

Kaydet x_{t}^{best}

END

Pheromone güncellemesi

Pheromone buharlaşması ($\tau_t = 0.1 * \tau_{t-1}$)

Pheromone izini güncelle($\tau_t = \tau_{t-1} + 0.01 * f(x_{\mathrm{t}-1}^{best})$)

Çözüm aşaması

Arama yönünü belirle ($\bar{x}_t^{best} = x_t^{best} + \left(x_t^{best} \times 0.01\right)$)

α vektörünün değerlerini üret

FOR *i*=1 TO *m*

Yeni koloninin değerlerini belirle ($X_t^k = x_{t-1}^{best} \pm dx$)

Yeni $f\left(x_{\mathrm{t}}^{best}\right)$ sonucunu belirle

Kaydet x_{t}^{best}

END

IF $f\left(\bar{x}_t^{best}\right) \leq f(x_t^{best})$ THEN $x^{global\ min} = (\bar{x}_t^{best})$

ELSE $x^{global\ min} = (\mathrm{x}_t^{best})$

END IF

$\alpha_t = \alpha_{t-1} * 0.99$

END

Şekil 3.4: Örnek Karınca Koloni Algoritması (Başkan ve diğ., 2009) .

3.3. PARÇACIK SÜRÜ OPTİMİZASYONU

Parçacık Sürü Optimizasyonu, parçacıkların sürü halinde hareket etmesinden yararlanan bir optimizasyon tekniğidir. (Kennedy ve Eberhart, 1995) tarafından, kuş ve balık sürülerinin iki boyutlu hareketlerinden esinlenilerek geliştirilmiş popülasyon tabanlı bir stokastik optmizasyon tekniğidir. Shi ve Eberhart (1998) tarafından optimizasyon modeline atalet ağırlığı eklenmiştir.

Parçacık Sürü Optimizasyonu, sosyal yaşamın basitleştirilmiş bir simulasyonudur. PSO'da, parçacıların herbiri bir çözüm adayını temsil eder. Bu parçacıklar bir optimizasyon probleminde çözüm uzayını araştırmak için kullanılırlar. Optimizasyon başlangıcında her bir parçacık rastgele ya da sezgisel olarak belirlenen bir konuma yerleştirilir ve sonraki adımlarda serbest harekete bırakılırlar. Her bir iterasyonda, her parçacık kendisinin ve çevresindekilerin uygunluğunu ölçer. Bu ölçümler bir sonraki iterasyonda parçacığın yeni konumunu bulmasında kullanılır.

Parçacık Sürü Optimizasyonu, genetik algoritmalar gibi evrimsel optimizasyon yöntemlerine benzetilmesine rağmen aralarında önemli farklar bulunmaktadır. Parçacık Sürü Optimizasyonu, parçacıkların uygun çözüme ulaşmak için evrime uğradığı popülasyon temelli bir algoritmadır.

Parçacık Sürü Optimizasyonunda parçacıklar kendilerinin geçmiş deneyimlerinden, uygunluk değerinin konumundan, komşularının geçmiş deneyimlerinden faydalanarak bir sosyal benzetim oluştururlar. Bir parçacığın komşusu belirlenirken; sürünün

tümüyle, belirli bir kısmıyla ya da sadece bir bireyle etkileşimini sağlayan komşuluk topolojilerinden faydalanılabilir. Parçacığın, sürü içersindeki diğer parçacıklarla etkileşimi o parçacığın yönünü bulmasında etkili bir unsurdur. Bu şekilde sürü içersindeki parçacıklar zamanla uygunluk değeri iyi olan parçacıklara yaklaşırlar ve muhtemel en iyi uygunluk bölgesini daha sıkı tararlar.

3.3.1. Komşuluk Topolojileri

Orijinal Parçacık Sürü Optimizasyonunda iki tür komşuluk türü vardır:

3.3.1.1. geniyi (gbest)

Global, en iyi anlamına gelmektedir ve bu topolojide sürü içersindeki bütün parçacıklar birbirinin komşusudur. İlk iterasyonda rastgele ya da sezgisel olarak dağıtılmış olan parçacıklar, tüm sürü ile haberleşirler ve en iyi uygunluk değerine sahip olan parçacık geniyi olarak seçilir. Sürüdeki diğer parçacıklar ise hareket yönlerini geniyiye göre yeniden belirlerler ve bir sonraki iterasyonda bu bilgi ışığında hareket ederler. Bu hareket sonucunda parçacıkların, arama uzayının en iyi kısmına doğru yönlendiği gözlemlenir ve hızla sonuca ulaşacağı düşünülebilir. Fakat optimum nokta geniyinin yakınında değilse, sürüdeki diğer parçacıkların geniyi etrafında toplanacağı için sürünün yerel optimum noktasında takılma ihtimali vardır.

3.3.1.2. leniyi (lbest)

Lokal en iyi anlamına gelmektedir. Bir parçacığın komşuları, sürü içersindeki belirli sayıda parçacıktan oluşmaktadır. Böylece tüm sürü içersindeki en iyi parçacık değil; parçacıklar için değişiklik gösterebilecek alt sürülerin en iyi parçacığına göre hız güncellemesi gerçekleştirilir. Daha yavaş sonuca ulaşmasına rağmen yerel optimuma takılma riski daha azdır.

Bu iki topolojide de parçacıklar arasındaki ilişkiler göz önünde bulundurularak yönlendirme yapıldığından iki topoloji de sosyal komşuluklar olarak görülebilir. Literatürde bulunan bir çok komşuluk topolojisi de burada bahsi geçen temelden yola çıkılarak türetilmiştir.

3.3.2. Temel Parçacık Sürü Optimizasyon Algoritması

Parçacık Sürü Optimizasyonunda, her bir parçacık sürü içersinde bulunan bir kuşu temsil eder. Kuşların yerini bilmedikleri bir yemi aramaları ise bir problemin çözümünü aramaya benzetilebilir. Yem arayışında olan kuşlar, yiyeceğe en çok yaklaşan kuşu takip eder. PSO'da ise her bir parçacık bulunduğu konumu uygunluk fonksiyonuna göndererek konumunun uygunluk değerini öğrenir. Her parçacık; konumunu, hızını, o ana kadar elde ettiği en iyi uygunluk değerini ve o uygunluk değerine sahip olan konumunu bilir. Bu bilgiler ışığında parçacık, diğer parçacıklarla iletişim kurar ve onların bilgileriyle kendisininkileri kullanarak bir sonraki konumunu belirler.

3.3.2.1. En Yakın Komşu Hızıyla Karşılaştırma ve Anlamsız Hareket Etme

Kennedy ve Eberhart (1995) tarafından en yakın komşu hızıyla karşılaştırma ve anlamsız hareket etme yöntemleri ile bir benzetim oluşturulmuştur. Bu benzetime göre, başlangıçta pozisyonları halka hücrelerden oluşan ızgaralarda bulunan bir sürü oluşturulur. Bu sürüdeki parçacıkların hızları X ve Y cinsinden belirlenir. Parçacıklar, hızları ve konumları rastgele olarak arama uzayına yerleştirilir. Her iterasyonda uygunluk değeri en iyi olan parçacığın X ve Y hızları incelemeyi yapan parçacığa atanır. Böylelikle sürü içersindeki parçacıklar arasında eş zamanlı bir hareket ortaya çıkar. Fakat parçacıların eş zamanlı olarak aynı yöne doğru hareket etmeleri ortak ve değişmeyen bir yere yerleşmelerine neden olur. Bu durumdan kurtulmak için "anlamsız hareket" (craziness) olarak adlandırılan stokastik bir değişken ilave edilmiştir. Her bir iterasyonda rastgele belirlenen X ve Y hızlarına bazı değişiklikler yapılarak sisteme yeterli değişkenlik kazandırılmıştır. Bahsedilen değişkenlik ise tamamen yapaydır.

3.3.2.2. Mısır Tarlası Vektörü

Benzetimin ikinci bölümü XY koordinatlarının iki boyutlu vektörü olan "mısır tarlası vektörünü" çıkartmıştır. Bu yönteme göre her parçacık konumunu bir denklem biçiminde ölçer. Bu ölçüme göre (100,100) pozisyonunda değer sıfırdır ve bu nokta global en küçük olarak belirlenir. Her parçacık en iyi uygunluk değerini ve bu değere ulaşmasını sağlayan konumunu hatırlar. Parçacığın en iyi uygunluk değeri $peniyi_n$, konumu ise $peniyix_n$ ve $peniyiy_n$ (n=1,2,…,N N=parçacık sayısı) ile ifade edilir. Her parçacık pozisyonları ölçerek hareket ederken, X ve Y yönündeki hızları ise basit bir

yöntem kullanılarak hesaplanır. Eğer parçacığın konumu, mısır tarlası üzerinde *peniyix*'in sağında kalıyorsa X hızı (*hx*) sistemde bulunan bir parametre ile ağırlıklandırılan rastgele bir değerle eksi yönde güncellenir. $hx_n^{t+1} = hx_n^t - \Phi \times p\_artış$ (Φ =0 ile 1 arasında rastgele bir sayı, t= iterasyon numarası). Eğer *peniyix*'in solunda kalıyorsa, $hx_n^t + \Phi \times p\_artış$ formülü ile güncellenir. Benzer biçimde parçacığın bulunduğu konumun *peniyiy*'nin altında ya da üstünde olması durumuna bağlı olarak hızı aşağı ya da yukarı doğru güncellenir. Ayrıca her bir parçacık, *geniyi* değerinden ve bu değere ulaşılan konumu ($peniyix_{geniyi}$, $peniyiy_{geniyi}$) bilir. Aynı biçimde her parçacığın hx_n^{t+1} *ve* hy_n^{t+1} değerleri *g_artış* sistem parametresi olmak üzere aşağıdaki biçimde güncellenir:

$$hx_n^{t+1} = \begin{cases} hx_n^t - \emptyset \times g\_artis & eger\ mevcutx_n > peniyix_{geniyi} \\ hx_n^t + \emptyset \times g\_artis & eger\ mevcutx_n < peniyix_{geniyi} \end{cases} \quad (3.3)$$

$$hy_n^{t+1} = \begin{cases} hy_n^t - \emptyset \times g\_artis & eger\ mevcuty_n > peniyiy_{geniyi} \\ hy_n^t + \emptyset \times g\_artis & eger\ mevcuty_n < peniyiy_{geniyi} \end{cases} \quad (3.4)$$

Simulasyon sonucunda *p_artış* ve g*_artış* değerleri nispeten yüksek ayarlandığında, sürü mısır tarlası içinde emiliyor gibi görünür. Birkaç iterasyon sonunda tüm sürü, en iyi uygunluk noktasını çevreleyen bir çember etrafında kümelenmiş gibi görünür. *p_artış* ve g*_artış* değerleri nispeten düşük ayarlandığında ise sürü, en iyi uygunluk noktası etrafında girdap benzeri bir görünüm oluşturarak yaklaşır. Bazı alt sürüler ritmik ve senkronize olarak sapma gösterir fakat sonunda onlar da hedefe ulaşırlar.

Temel PSO algoritmasının ikili ve sürekli olmak üzere iki türü bulunur:

3.3.2.3. İkili Parçacık Sürü Optimizasyonu

İkili PSO Kennedy (1997) tarafından geliştirilmiştir. Burada parçacık değerleri 0 ya da 1 olarak belirlenir.

$$mevcut_{nd}^{t+1} = \begin{cases} 1 \; eğer \; \emptyset_3 < \frac{1}{1+e^{-h_{nd}^{t+1}}} \\ 0 \quad değilse \end{cases} \tag{3.5}$$

ya da

$$mevcut_{nd}^{t+1} = yuvarla\left\{\frac{1}{1+e^{-(mevcut_{nd}^{t}+h_{nd}^{t+1})}} - 0.06\right\} \tag{3.6}$$

3.3.2.4. Sürekli Parçacık Sürü Optimizasyonu

Sürekli Parçacık Sürü Optimizasyonu, reel değerlerden oluşan çok boyutlu bir çözüm uzayı kullanır. Bu çözüm uzayındaki her bir parçacık pozisyonunu Denklem (3.7) ve (3.8) ile belirler.

$$h_{nd}^{t+1} = \omega \times h_{nd}^{t} + c_1 \times \emptyset_1 \times (peniyi_{nd}^{t} - mevcut_{nd}^{t}) + c_2 \times \emptyset_2 \times (geniyi_{d}^{t} - mevcut_{nd}^{t}) \tag{3.7}$$

$$mevcut_{nd}^{t+1} = mevcut_{nd}^{t} + h_{nd}^{t} \tag{3.8}$$

h_{nd}^{t} : $t.$ iterasyonda $n.$ parçacığın d değişkeni için hızı

ω : Dinginlik katsayısı

$mevcut_{nd}^{t}$: $t.$ iterasyonda $n.$ parçacığın d değişkeninin değeri

c_1, c_2 : Ağırlıklandırma sabiti

$\emptyset_1, \emptyset_2$: $0-1$ aralığında rastgele değer

$peniyi_{nd}^{t}$: $t.$ iterasyona kadar $n.$ parçacığın en iyi d değişken değeri

$geniyi_{d}^{t}$: $t.$ iterasyona kadar komşu olarak belirlenen parçacıklar içersinde en iyi d değişken değeri

$geniyi_d$ değişken değeri hesaplanırken kullanılan komşular, komşuluk türüne bağlı olarak belirlenir. Temel PSO algoritmasında global (geniyi) ve yerel (leniyi) çevre kullanılır. Global çevrede $geniyi_d$ belirlenirken sürüdeki tüm parçacıklar kullanılır. Yerel çevrede ise sürü içersindeki belirli bir grup ele alınır. Parçacığın yerel komşuları belirlendikten sonra algoritmanın sonuna kadar değiştirilmez (Cura, 2008b).

h^t_{nd}'i sınırlandırarak yakınsamayı sağlamak için bir h_{max} sabiti kullanılabilir. h_{max} sabitinin değeri genel olarak [$-mevcut^{max}_{nd}$, $mevcut^{max}_{nd}$] aralığındadır (Eberhart ve Shi,

1998, Clerc ve Kennedy, 2002) hız üzerindeki sınırlandırma ihtiyacından dolayı denklem (3.9) ‘u ortaya atmıştır (Bali, 2009).

$$h_{nd}^{t+1} = \alpha \times \begin{bmatrix} h_{nd}^{t} + c_1 \times \emptyset_1 \times \left(peniyi_{nd}^{t} - mevcut_{nd}^{t}\right) + \\ c_2 \times \emptyset_2 \times \left(geniyi_{d}^{t} - mevcut_{nd}^{t}\right) \end{bmatrix} \tag{3.9}$$

$$\alpha = \frac{2}{\left|2 - c - \sqrt{c^2 - 4 \times c}\right|} \quad c = c_1 + c_2 \qquad c > 4 \tag{3.10}$$

Genel Parçacık Sürü Optimizasyon algoritması Şekil 3.5’te verilmiştir.

```
Parametreleri Belirle
Başlangıç Popülasyonunu Oluştur
Her bir Parçacık için{
   Amaç Fonksiyon Değerlerini Bul
   Yerel En iyiyi Bul
   }
Global En iyiyi Bul
Yap{
   Her bir Parçacık için{
   Hız Vektörünü Güncelle
   Pozisyon Vektörünü Güncelle
   Amaç Fonksiyon Değerlerlerini Bul
   Yerel En iyiyi Güncelle
   }
   Global En iyiyi Güncelle
Durdurma Kriteri
```

Şekil 3.5: Örnek Parçacık Sürü Optimizasyon Algoritması (Bali, 2009).

3.4. BENZETİLMİŞ TAVLAMA

Benzetilmiş Tavlama (Simulated Annealing) metallerin soğutularak minimum enerjili kristal yapıya dönüşmesi ile bir problemde minimumu arama işlemi arasında benzerlik

kurarak optimizasyon problemlerine çözüm arar. Benzetilmiş Tavlama, Metropolis ve arkadaşlarının (Metropolis ve diğ., 1953) belirli bir ısı seviyesinde atomların dengeli dağılımını bulma amacıyla geliştirdikleri ve enerji değişimlerini taklit eden bir çalışmayı temel almaktadır. Benzetilmiş Tavlamanın bir optimizasyon tekniği olarak kullanılabilmesi fikri ise (Kirkpatrick ve diğ., 1983) tarafından ortaya atılmıştır. Benzetilmiş Tavlama yaklaşımı, uygulama şeklinin basit olması ve geniş bir uygulama alanının bulunması sebebiyle birçok araştırmacı tarafından ilgi görmüştür.

Benzetilmiş Tavlama metodunun temelinde yatan tavlama işlemi, katı bir maddenin gücünü artırmak için erime noktasını aşıncaya kadar ısıtılıp ardından soğutulması işlemi olarak tanımlanabilir. Soğuma hızına bağlı olarak katı maddenin yapısal özellikleri de değişiklik gösterir.

Benzetilmiş Tavlamanın diğer yöntemlerden en önemli farkı yerel minimum noktasına takılmama yeteneğidir. Bu yetenek Benzetilmiş Tavlamanın amaç fonksiyonundaki kötüye gitme durumunu da kabul edebilmesinden gelmektedir. En iyiyi arama işleminde ulaşılan yeni pozisyonda eğer amaç fonksiyonunda kötüleşme varsa bir kabul eşiği olasılığına bağlı olarak yeni pozisyon seçilir ya da bir önceki pozisyona geri dönülür. Benzetilmiş Tavlama yapısı itibari ile yerel komşu arama tekniği ile benzerlik gösterir. Yerel arama işleminde; mevcut çözüme, komşu çözümler içinde sürekli denemeler yaptırılır. Bu teknik başlangıç noktasına bağlı olarak sıklıkla yerel en iyi noktasına yönelmektedir. Benzetilmiş Tavlama ise kötüye giden çözümleri de araştırabildiğinden yerel en iyiden çıkarak global en iyiye ulaşabilmektedir.

3.4.1. Yerel Arama Algoritması

Gerçek hayatta bir minimizasyon probleminde, çözüm kümesi çok büyük bir küme olabilir. Bu durumda çözüm kümesi içersinden bir alt küme seçilerek çözüm araştırılır. Böyle durumlarda ise yerel arama algoritmaları kullanılır.

Yerel Arama Algoritması çözüm kümesinin alt kümesinde bir nokta belirler ve bu noktanın komşularını arar. Eğer aranan komşu çözümün uygunluk değeri, mevcut çözümün uygunluk değerinden küçükse komşu çözüm mevcut çözüm olarak belirlenir.

3.4.2. Eşik Algoritmaları

Eşik algoritmalarında, yerel arama algoritmasında olduğu gibi bir başlangıç noktasından yola çıkılarak komşular araştırılır. Yerel arama algoritmasından farklı olarak; komşu çözümün uygunluk değeri mevcut çözümün uygunluk değerinden küçükse kabul edilir, büyükse aralarındaki fark bir eşik değeri ile karşılaştırılır. Eğer aralarındaki fark eşik değerinden küçükse komşu çözüm mevcut çözüm olarak kabul edilebilir. Genellikle eşik değeri bir olasılık belirleyici olarak kullanılır. Komşu çözümle mevcut çözüm arasındaki fark, eşik değerinin ne kadar altındaysa kabul edilme olasılığı da o kadar fazladır. Fakat gene de kabul edilme garantisi yoktur. Bununla birlikte iterasyon sayısı arttıkça eşik değeri de azaltılır.

3.4.3. Tavlama İle Benzerlik

Tavlama termal bir işlemdir ve enerjinin düşük olduğu durum katı hali temsil eder. Buna göre tavlama, katı maddenin asgari erime noktasına kadar ısıtılması ve ısıtılmış maddenin yavaşça soğutularak katı haline döndürülmesi olmak üzere iki adımda incelenebilir. Sıvı halde maddelerin parçacıkları rastlantısal olarak dağılır. Katı halde ise kuvvetli bir yapısal bütünlük içindedir.

Metropolis ve diğ. (1953) tavlama sürecinin benzetimi için basit bir algoritma geliştirmiştir. Temelde Monte Carlo tekniklerine dayanan algoritma, maddenin mevcut halindeki enerji değerini bir sonraki halinin enerji değerinden çıkartır ve sonuç değer sıfırdan küçük veya eşitse yeni hali kabul eder. Aksi durumda ise bir olasılık denklemine göre kabul durumunu yeniden inceler.

Metropolis ve diğ. (1953) ortaya atmış olduğu algoritma optimizasyon problemlerine uydurulduğunda madde bir optimizasyon problemini, maddenin halleri ise farklı çözümleri temsil etmektedir. Böylece mevcut haldeki enerji, çözümün uygunluk değerine karşılık gelecektir. Tavlama işleminde madde en düşük enerji seviyesine ulaşıncaya kadar soğutulur. Böylelikle optimizasyon problemi için minimum noktaya ulaşılmış olur.

3.4.4. Tavlama Algoritması

Genel yapısı itibariyle Benzetilmiş Tavlama, yerel arama algoritmasına benzese de kötü sonuçlara da izin vererek yerel en iyiye takılmadığı için bir eşik algoritması türü olarak kabul edilebilir.

Genel Benzetilmiş Tavlama algoritması Şekil 3.6'da verilmiştir.

i başlangıç çözümü rastlantısal olarak belirlenir

t_0 başlangıç ısısı belirlenir

k=0

while $(t_k > t_{son})$ {

    For *c=1* to *n* {

        j komşu çözümü, i mevcut çözümünün komşuları arasından rastlantısal olarak belirlenir

        if ($r \leq P_{tk}${*j'yi kabul et*}) *i=j*

    }

    k=k+1

    $t_k = \alpha(t_k - 1)$ *//ısı düşürme fonksiyonu*

}

Şekil 3.6: Örnek Benzetilmiş Tavlama Algoritması

Algoritmadaki r değişkeni, her iterasyonda [0,1] aralığında rastgele seçilen bir sayıdır. n değişkeni problemin yapısına göre kullanıcı tarafından belirlenen araştırılacak komşu sayısıdır. Algoritmada geçen P_{tk}{j'yi kabul et} fonksiyonu ise Denklem (3.12)'ye göre hesaplanır.

$$\mu = \frac{f(i)-f(j)}{t_k} \tag{3.11}$$

$$P_{tk}\{j'yikabul\ et\} = \begin{cases} \min(e^{\mu}, 1)\ eğer\ f(j) > f(i), \\ 1\ değilse \end{cases} \tag{3.12}$$

i arama uzayı içersindeki mevcut çözümü, j ise i mevcut çözümünün aranan komşusu olarak ifade edilir. Burada eğer i mevcut çözümünün uygunluk değeri j komşu çözümünün uygunluk değerinden küçükse j çözümü kabul edilir. Aksi durumda ise k iterasyonundaki eşik değerine bağlı olarak değişen bir olasılıkla kabul edilir (Cura, 2008b).

3.5. TABU ARAMA ALGORİTMASI

Tabu Arama Algoritması, Glover (1986) tarafından ortaya atılan bir arama algoritmasıdır. Temel olarak Tabu Arama Algoritması; son çözüme götüren adımın, doğrusal hareketleri engellemek için bir sonraki iterasyonda tekrar kullanımının yasaklanması veya cezalandırılması şeklinde çalışmaktadır. Tabu arama, daha önce incelenmiş çözümlerin oluşturduğu yol üzerindekiler hariç her çözümü inceleyebilen bir tekniktir. Bu sayede yeni çözümler incelenerek yerel minimum noktasından kaçınılmış olur.

Algoritma temel olarak önce yerel minimum noktasına doğru hareket eder. Önceki hareketlerin tekrarlanmaması için bir veya birden fazla tabu listesi oluşturarak bu listeyi sürekli kontrol eder. Tabu listesinin asıl amacı önceden yapılmış hareketlerin tekrarı değil, ters yönde hareket yapılarak başa dönmeyi önlemektir. Tabu listesinin rolü, algoritma ilerledikçe değişiklik gösterebilir. Başlangıçta çözüm uzayında kaba araştırma yapılırken aday konumlar belirlendikçe arama yerel minimum noktalarına odaklanır (Cura, 2008b).

Bir optimizasyon probleminde çözüm, çözüm uzayında aranmaktadır. Tabu Arama Algoritmasında, tabu listesinde tanımlanan yasaklı noktalar incelenmemektedir. Bir noktanın tabu listesine girebilmesi ise daha önce o noktanın incelenmiş olması ve tekrar incelenmesine gerek olmaması ile mümkündür. Bu duruma ek olarak tabu listesi,

istenmeyen noktalar listesi olarak kullanıldığı gibi aynı zamanda çözüme götüren daha iyi noktaların tutulduğu bir liste olarak da kullanılabilir. Böylelikle iyi çözümler işaretlenerek arama algoritmasının o bölgelere odaklanması sağlanabilir.

Tabu arama algoritmasına kullanılan liste dört ana ilkeye bağlı olarak belirlenir: Yakın geçmişte incelenmiş noktalar, çözümün incelenme sıklığı, çözümün kalitesi ve çözümün etkisi (Cura, 2008b).

Bir optimizasyon probleminde çözüm uzayı araştırılırken tabu listesine başvurulur ve tabu listesinde bulunan bir hareket vektörü 1 veya 0 değerini alabilmektedir. Bu durumda 1 olarak işaretlenmiş hareket yapılırken 0 olarak işaretlenen hareket yapılmamaktadır (Cura, 2008b). Bahsi geçen hareket olgusu mevcut çözümün bir komşusunun seçilmesini temsil etmektedir. Tabu listesi, komşu seçiminde kullanılan kuralları barındırdığı için komşu arama algoritması olarak düşünülebilir.

3.5.1. Komşu Arama Algoritması

Optimize edilmesi gereken bir problemde tüm çözüm kümeleri incelenmelidir; fakat gerçek hayatta bu durum mümkün olamayacağından alt çözüm kümeleri araştırılmaktadır. Komşu arama algoritmasında problemin alt çözüm kümesi içersinden bir nokta belirlenir. Bir komşu seçme algoritması ile komşular mevcut noktanın yerine geçer. Tabu arama algoritmasında ise tek bir komşu seçme fonksiyonu olmadığından genel komşu arama algoritmalarından farklı bir bitirme koşuluyla sonlandırılmaktadırlar. Orijinal komşu arama algoritmaları alt çözüm kümelerindeki tüm komşular incelendiğinde sonlanmaktadır. Tabu Arama Algoritmasında ise bir iterasyon sayısı sonlandırma koşulu olarak kullanılabileceği gibi komşu çözümün mevcut çözümden kötü olma durumu da kullanılabilir. Algoritma geliştiricileri sonlandırma koşulunu belirlerken problemin yapısına göre karar vermektedirler.

3.5.2. Temel Tabu Arama Algoritması

Tabu Arama Algoritmasında kullanılan tabu liste yapılarından en çok kullanılanı yakın geçmişte yapılan hareketin tekrarlanmasına dayalı olandır. Bu yapıda tabu süresi

bellekte tutulur. Bahsedilen tabu süresi iterasyon sayısını temsil etmektedir ve iterasyon boyunca aynı hareketin tekrarlanmasını engellemek için kullanılır.

Tabu listesinde bulunan ve içinde tabu sürelerini de barındıran hareket vektörleri *h* ile ifade edilirse başlangıçta hiçbir hareket gerçekleşmediği için her hareketin listedeki karşılığı h_x değeri 0'a eşitlenir. Tabu listesinden herhangi bir hareket seçildiğinde ise seçilen harekete karşılık gelen h_x değeri tabu süresinin bir fazlası ile mevcut iterasyonun *(t)* toplamına eşitlenir. Bazen *x* hareketinin seçilmesi, tabu olarak işaretlenmesi için yeterli olmamaktadır. Aynı zamanda *x* hareketinin daha iyi bir çözüme gitmesi de gerekmektedir. Bir hareket seçileceği zaman tabu listesindeki değeri *t*'ye küçük eşit olan hareketler seçilir.

Tabu süresi ε ile ifade edildiğinde tabu süresinin tüm hareket sayısından (X) küçük (ε<X) olması gerektiği görülmektedir. Aksi takdirde tüm aday çözümlere bakıldıktan sonra hiçbir hareket seçilemeyecektir.

Örnek bir Tabu Arama Algoritması Şekil 3.7'de verilmiştir.

$h_x=0 \quad \forall x \in X$

bitme_koşulu=yanlış

t=0

do

$x \in X$ ve $h_x \leq t$ olan *x* hareketi rastlantısal olarak seçilir

$j=\Psi^x(i)$

$h_x=t+\varepsilon+1$

if *(f(j)<f(i))i=j //çözüm daha iyiyse komşu çözümü mevcut çözüm olarak kabul et*

if *(bitme koşulu sağlanıyor) bitme_koşulu=doğru*

t=t+1

while *bitme_koşulu≠doğru*

Şekil 3.7: Temel Tabu Arama Algoritması

3.5.3. Tabu Arama Bellek Yapıları

3.5.3.1. Yakın Geçmişe Dayalı Bellek Yapısı

Yakın Geçmişe Dayalı Bellek, en temel tabu bellek yapısı olup görevi yakın geçmişte gerçekleştirilen hareketlerin tekrarını bir süreliğine tabu olarak işaretlemektir. Bu bellek yapısında çok bilinen iki uygulama vardır. İlki sadece ters hareketlerin tabu olarak işaretlenmesidir. Bunun için $tabu\_başla_x$ ve $tabu\_bitir_x$ şeklinde iki dizi kullanılabilir. Burada x komşu seçme fonksiyonunu yani hareketi temsil etmektedir. $tabu\_başla_x$, x hareketinin tabu olarak işaretlendiği iterasyon numarasını ve $tabu\_bitir_x$ de x hareketinin tabu olmaktan çıktığı iterasyon numarasını göstermektedir. Tabu süresi ise (ε)= $tabu\_bitir_x$ - $tabu\_başla_x$ olacaktır.

Hareket olarak adlandırılan komşu seçme fonksiyonu $j=\Psi(i)$ ile gösterilir. Bu hareket vektörü i çözümünden j çözümüne giden yöndedir. Hareket vektörünün tersi ise j çözümünden i çözümüne olan $i=\Psi^{-1}(j)$ 'dir. Bu yöntemde i çözümünden j çözümüne hareket gerçekleştiğinde hareket yönünün tersi olan j çözümünden i çözümüne doğru olan hareket tabu olarak işaretlenecektir. Hareketin kendisi ise tabu olmamaktadır.

3.5.3.2. Sıklığa Dayalı Bellek Yapısı

Bu bellek yapısı, yakın geçmişe dayalı bellek yapısını tamamlayıcı niteliktedir ve genellikle yakın geçmişe dayalı bellek yapısı ile birlikte ikinci bir bellek olarak kullanılır. Sıklığa dayalı bellek yapısı gerçekleştirilen hareketlerin sıklığını hafızada tutar. Bir hareketin kaç kez gerçekleştirildiği bilgisi ile birlikte o hareketin sonucu olan çözümün kalitesi ve hareketin sonuçlarına bağlı diğer bilgilerin tutulması daha yararlı

bir yaklaşım olacaktır. Sıklık ölçütünün dört bileşeni; her bir hareketin tekrar sayısı, toplam hareket sayısı, en yüksek tekrar sayısı ve ortalama tekrar sayısıdır.

Her bir hareketin tekrar sayısı, bir hareket gerçekleştiğinde o hareketin daha önce kaç defa gerçekleştiği bilgisini taşır. Fakat hareketin kalitesini artırmak için mevcut çözümün sonucundan daha kötü sonuca götüren hareketlerde sayaç değeri azaltılabilmektedir. Bu sayede toplam hareket sayısı, en yüksek tekrar sayısı ve ortalama tekrar sayısı da negatif değer alabilmektedir. Negatif olmaları durumu da sonlandırma koşulu olarak kullanılabilir.

Sıklığa Dayalı Bellek Yapısı, yakın geçmişe dayalı bellek yapısından farklı olarak yapılan hareketin tabu olarak işaretlenmesinden çok, iyi hareketlerin tekrarlanmasını amaçlar. Her hareketin seçilme olasılığı; tekrar sayısının toplam hareket sayısına, en yüksek tekrar sayısına ve ortalama tekrar sayısına bölünmesiyle elde edilen sonuca bağlıdır. Bu şekilde önceden daha iyi sonuca götüren hareketlerin yapılma olasılığı artmış olacaktır (Cura, 2008b).

4. MATERYAL YÖNTEM

4.1. METASEZGİSEL YÖNTEMLERİN DOĞRULUĞUNUN SONUCU BİLİNEN PROBLEMLERLE TEST EDİLMESİ

Kullanılan meta sezgisel yöntemlerin algoritmalarının doğruluğunun test edilebilmesi için sonucu önceden bilinen iki test fonksiyonu kullanılmıştır.

4.1.1. Branin Fonksiyonu

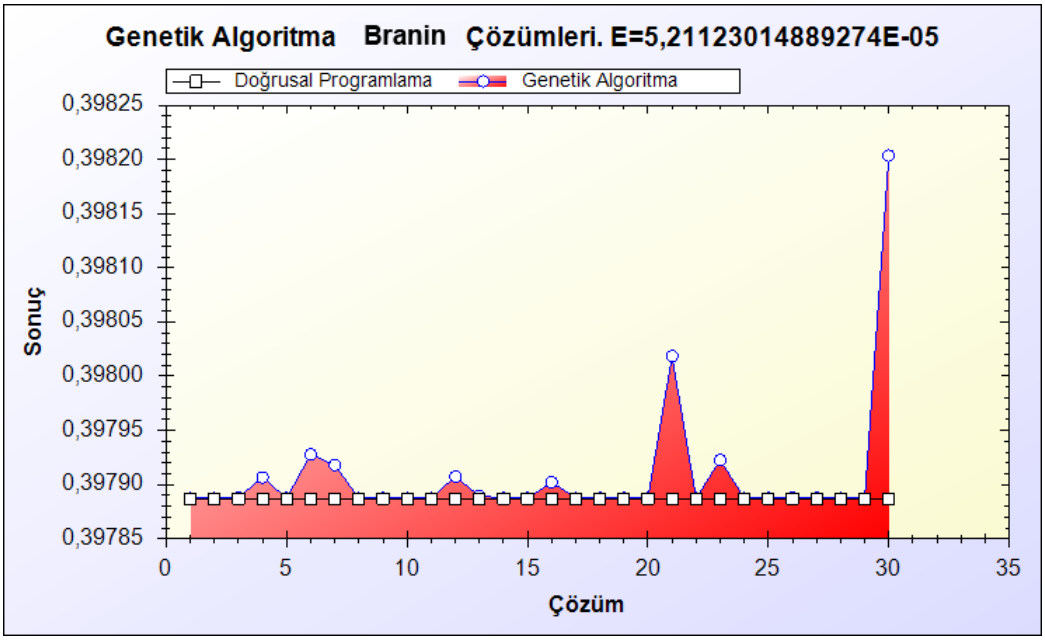
Değişken sayısı n=2,

$$\text{Optimal } f(x) = (x_2 - \frac{5.1}{4\pi^2}x_1^2 + \frac{5}{\pi}x_1 - 6)^2 + 10\left(1 - \frac{1}{8\pi}\right)\cos(x_1) + 10 \qquad (4.1)$$

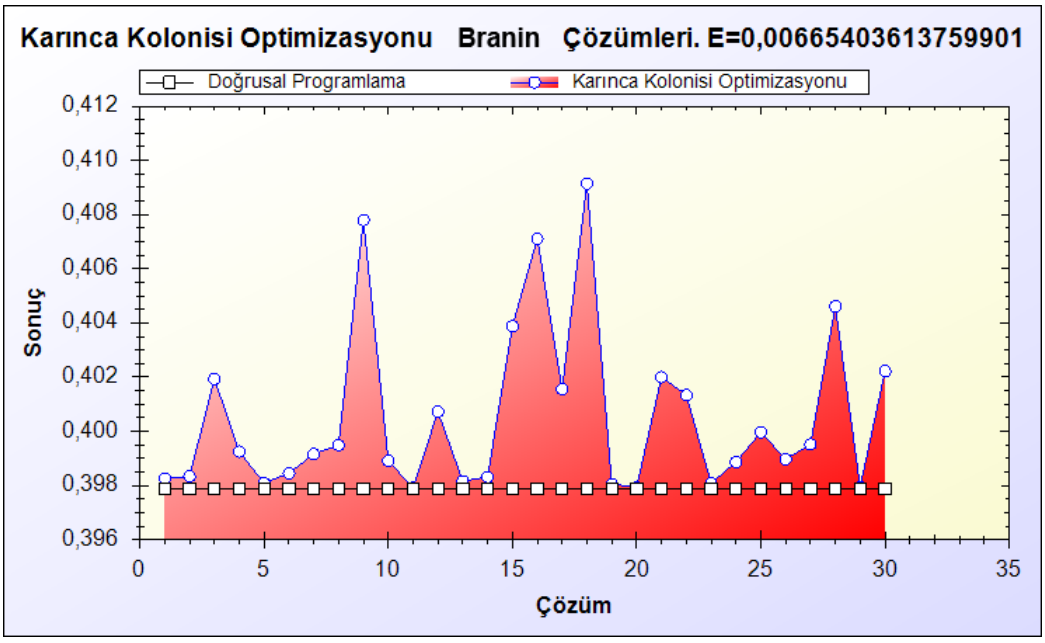
Kısıtlar $-5 \leq x1 \leq 10$, $0 \leq x2 \leq 15$

Sonuçlar x* = (-π , 12.275), (π , 2.275), (9.42478, 2.475), f(x*) = 0.397887

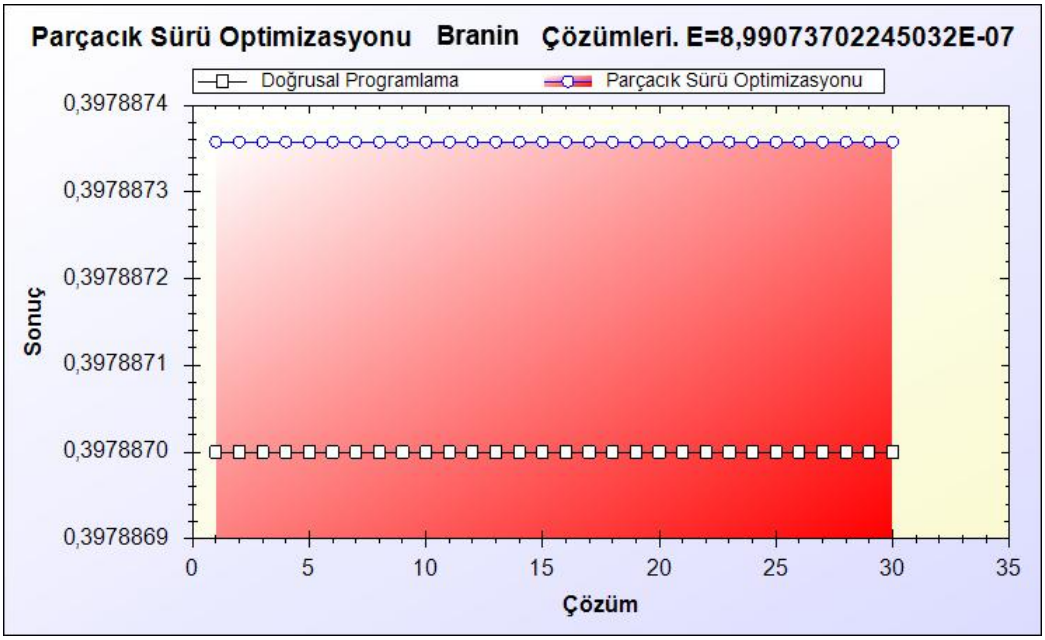
Aşağıdaki grafiklerde Branin fonksiyonunun meta sezgisel yöntemlerle çözümünden elde edilen grafikler görünmektedir. Fonksiyon her bir meta sezgisel yöntemle 30 kez çözdürülmüş olup bu 30 çözümün sonuçları her bir meta sezgisel yöntem için ayrı ayrı grafiksel olarak gösterilmektedir. Grafiklerdeki kare şeklindeki noktalar Branin Fonksiyonunun gerçek sonucunu daire şeklindeki noktalar ise meta sezgisel yöntemle elde edilen sonucu ifade etmektedir.



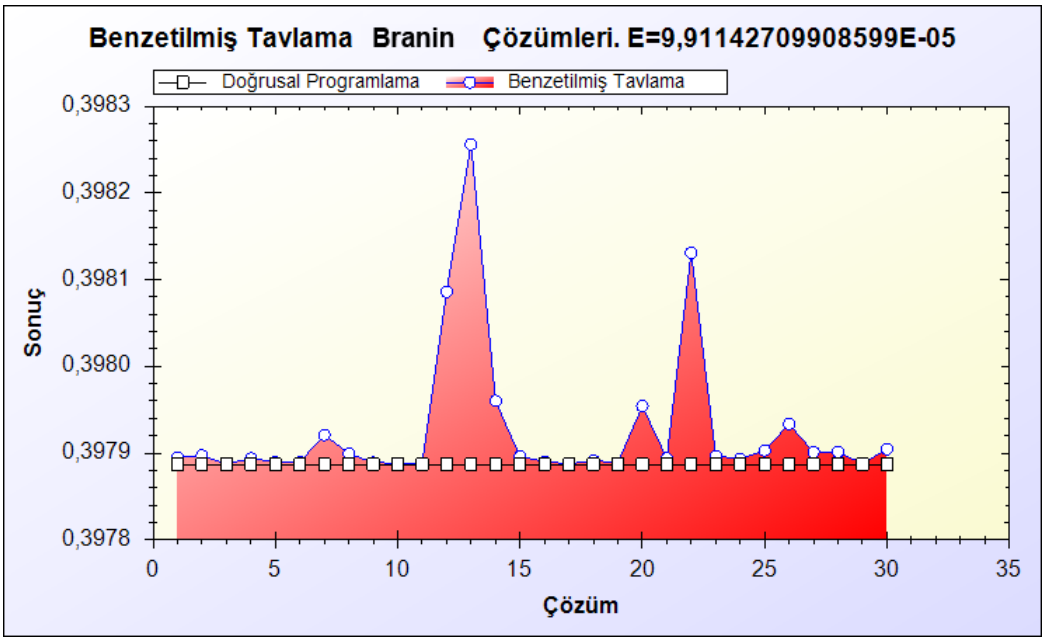
Şekil 4.1: Branin Fonksiyonu Genetik Algoritma Çözüm Grafiği



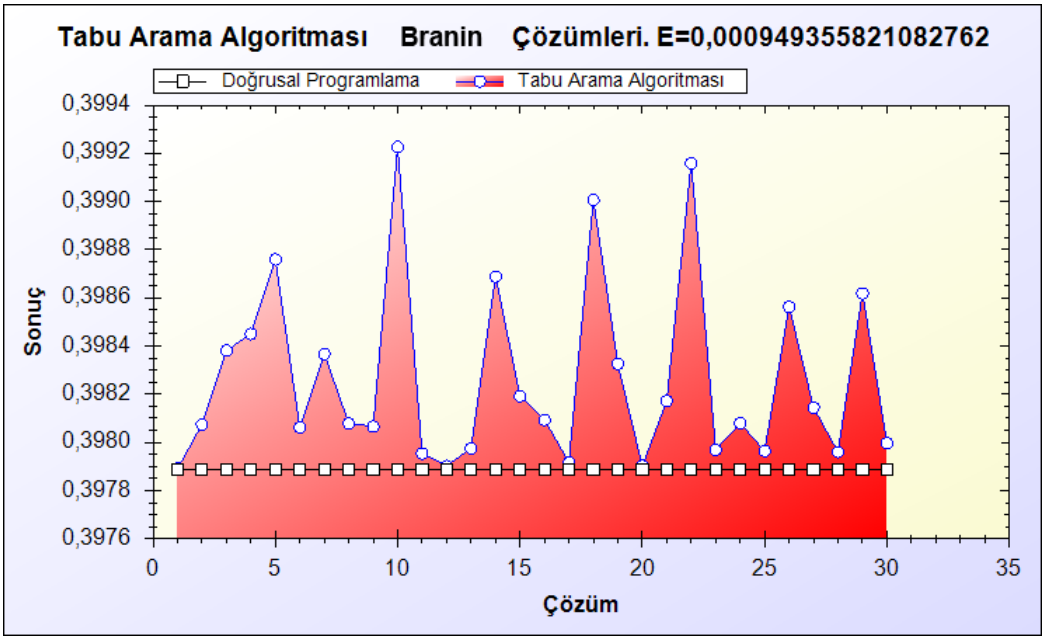
Şekil 4.2: Branin Fonksiyonu Karınca Kolonisi Optimizasyonu Çözüm Grafiği



Şekil 4.3: Branin Fonksiyonu Parçacık Sürü Optimizasyonu Çözüm Grafiği



Şekil 4.4: Branin Fonksiyonu Benzetilmiş Tavlama Çözüm Grafiği



Şekil 4.5: Branin Fonksiyonu Tabu Arama Algoritması Çözüm Grafiği

Yukarıdaki grafiklerden elde edilen sonuçlar Çizelge 4.1'de sunulmuştur. Çizelgeden de anlaşılacağı üzere maksimum binde 6,7 gibi bir hata oranıyla doğru çözüme ulaşılmıştır. Çizelge üzerinde, gerçekleştirilen 30 çözümün ortalaması alınıp ortalamaya sayısal olarak en yakın çözümün sonuçları "Ortalamaya En Yakın Sonuç" ve bu sonuca ulaşılırken harcanan süre "Süre", çözümler içersinde bulunana en iyi sonuç "Minimum Sonuç" ve bu sonuca ulaşırken harcanan süre "Süre" başlığı altında verilmektedir.

Çizelge 4.1: Branin Fonksiyonunun Meta Sezgisel Yöntemlerle çözümünden elde edilen değerler.

Meta Sezgisel Yöntem	Bağıl Hata	Ortalamaya En Yakın Sonuç	Süre(sn)	Minimum Sonuç	Süre(sn)
GA (Matlab)	5,21E-05	0,397907	0,66346	0,397887	0,794772
KKO	0,006654	0,400729	12,53337	0,397917	12,65889
PSO	8,99E-07	0,397887	4,942356	0,397887	4,942356
BT (Matlab)	9,91E-05	0,397921	1,164352	0,397887	1,365085
TA	0,000949	0,398326	8,241297	0,397891	8,560281

4.1.2. Sphere Fonksiyonu

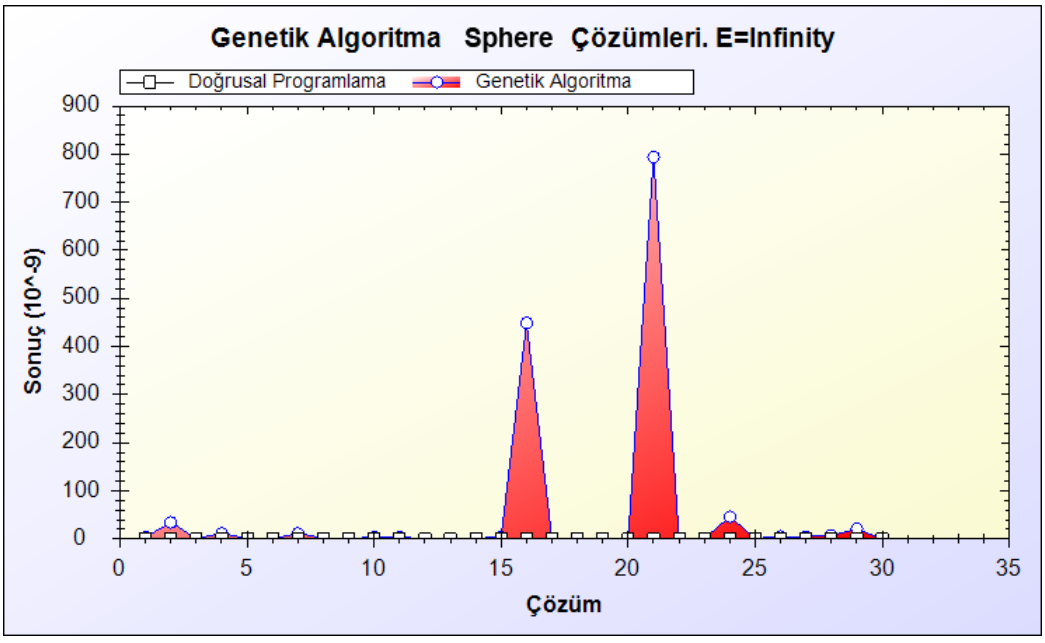
Değişken sayısı n=2,

Optimal $f(x) = \sum_{i=1}^{n} x_i^2$ (4.2)

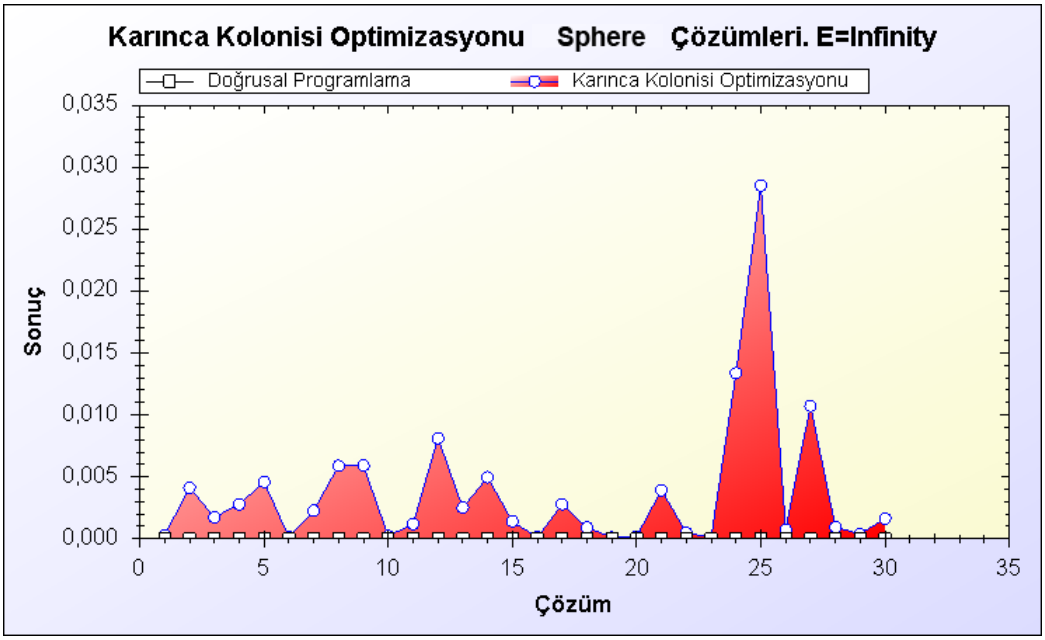
Kısıtlar $-5.12 \leq x_i \leq 5.12$, i = 1, 2, . . . , n.

Sonuçlar x* = (0, …, 0), f(x*) = 0

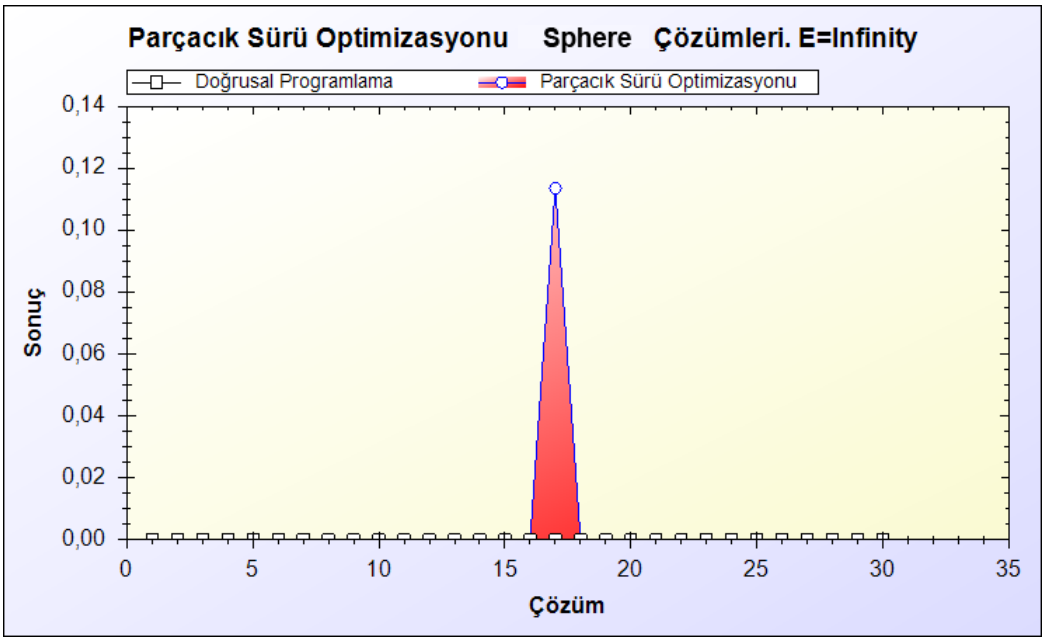
Aşağıdaki grafiklerde Sphere fonksiyonunun meta sezgisel yöntemlerle çözümünden elde edilen grafikler görünmektedir. Fonksiyon her bir meta sezgisel yöntemle 30 kez çözdürülmüş olup bu 30 çözümün sonuçları her bir meta sezgisel yöntem için ayrı ayrı grafiksel olarak gösterilmektedir. Grafiklerdeki kare şeklindeki noktalar Sphere Fonksiyonunun gerçek sonucunu daire şeklindeki noktalar ise meta sezgisel yöntemle elde edilen sonucu ifade etmektedir.



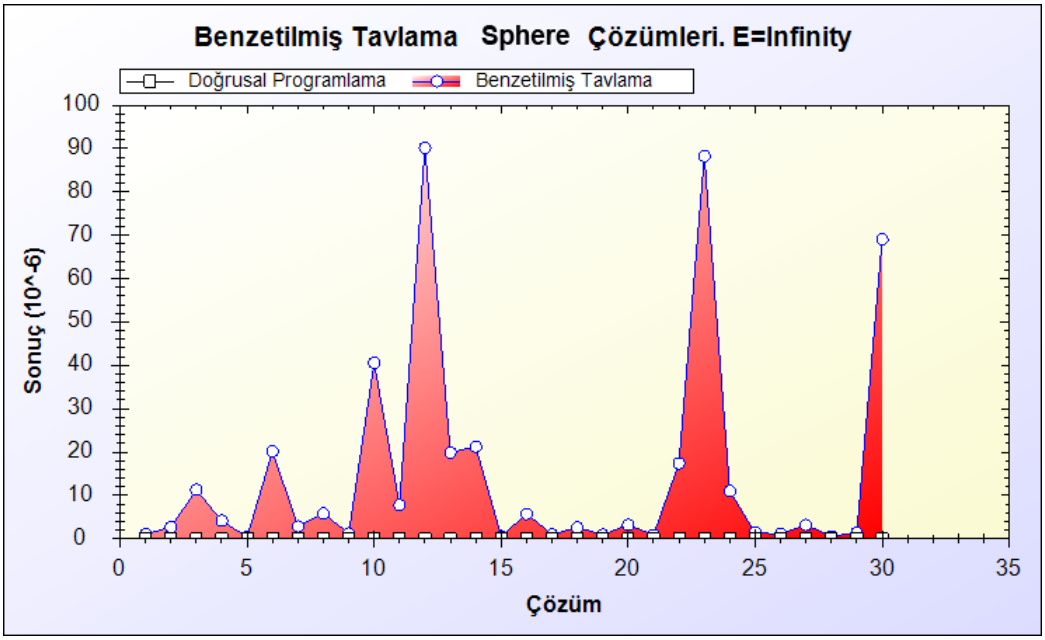
Şekil 4.6: Sphere Fonksiyonu Genetik Algoritma Çözüm Grafiği



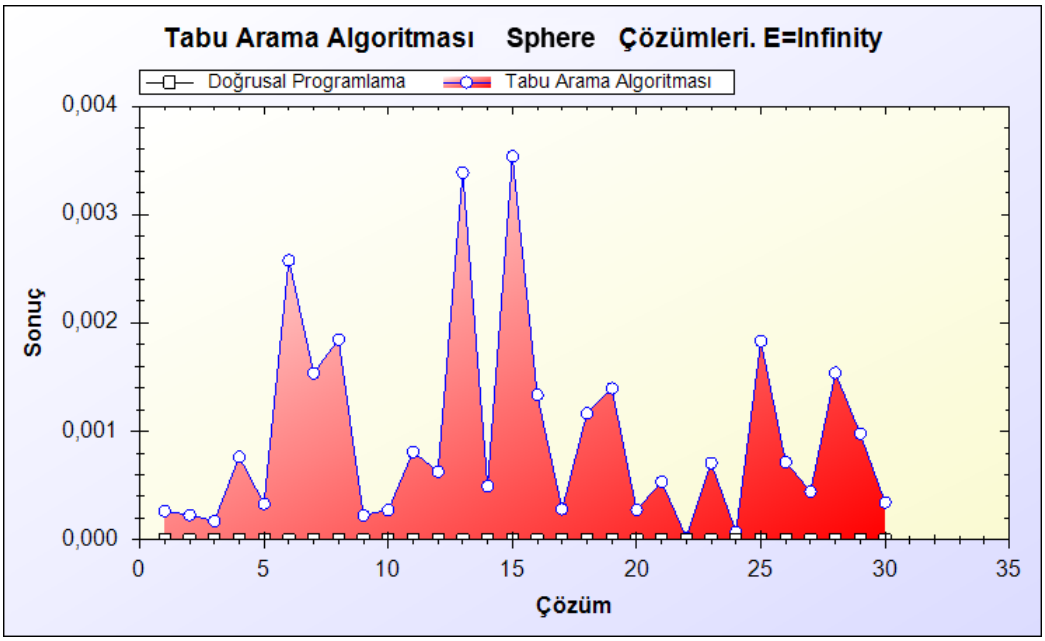
Şekil 4.7: Sphere Fonksiyonu Karınca Kolonisi Optimizasyonu Çözüm Grafiği



Şekil 4.8: Sphere Fonksiyonu Parçacık Sürü Optimizasyonu Çözüm Grafiği



Şekil 4.9: Sphere Fonksiyonu Benzetilmiş Tavlama Çözüm Grafiği



Şekil 4.10: Sphere Fonksiyonu Tabu Arama Algoritması Çözüm Grafiği

Yukarıdaki grafiklerden elde edilen sonuçlar Çizelge 4.2'de tablo sunulmuştur. Çizelge üzerinde, gerçekleştirilen 30 çözümün ortalaması alınıp ortalamaya sayısal olarak en yakın çözümün sonuçları "Ortalamaya En Yakın Sonuç" ve bu sonuca ulaşılırken harcanan süre "Süre", çözümler içersinde bulunana en iyi sonuç "Minimum Sonuç" ve bu sonuca ulaşırken harcanan süre "Süre" başlığı altında verilmektedir. Ortalamaya En

yakın Sonuçlarda maksimum binde 3'lük bir farkla sonuca yaklaşılmış, minimum sonuçlarda ise en fazla $8X10^{5}$'lik bir farkla sonuca yaklaşılmıştır.

Çizelge 4.2: Sphere Fonksiyonunun Meta Sezgisel Yöntemlerle çözümünden elde edilen değerler.

Meta Sezgisel Yöntem	Ortalamaya En Yakın sonuç	Süre(sn)	Minimum Sonuç	Süre(sn)
GA (Matlab)	4,62E-08	0,788179	4,21E-11	0,642934
KKO	0,003929	13,69875	8,55E-05	13,69371
PSO	2,69E-35	1,308729	1,02E-37	1,291446
BT (Matlab)	1,74E-05	1,689962	3,70E-07	2,228772
TA	0,00098	1,346568	2,14E-05	1,3624

Grafikler ve çizelgelerde de görüldüğü gibi meta sezgiseller için kullanılan algoritmalar test için kullanılan Branin ve Sphere Fonksiyonlarında başarılı sonuçlar vermektedir.

4.2. DOĞRUSAL PROGRAMLAMA PROBLEMLERİNİN TANIMLANMASI

Meta sezgisel yöntemlerin karşılaştırılması için kullanılan doğrusal programlama problemleri rastgele olarak üretilmiş 20 problemden oluşmaktadır. Her problem kendi aralarında farklı değişken sayısı ve kısıt şartına sahiptir.

Problem 1:

$Z_{1min} = 7x_1 + 5x_2$

Kısıtlar:

$2x_1 + 3x_2 \leq 20$

$8x_1 + 5x_2 \geq 15$

$15x_1 + 49x_2 \leq 105$

Doğal Kısıtlar:

$x_{j\geq 0}$, j=1,2

Problem 2:

$Z_{2min} = -7x_1 + 5x_2 + 3x_3$

Kısıtlar:

$2x_1 + 4x_2 \leq 30$

$5x_1 + 9x_2 + 7x_3 \leq 50$

$4x_1 + 7x_2 + 13x_3 \leq 170$

Doğal Kısıtlar:

$x_{j\geq 0}$, j=1,2,3

Problem 3:

$Z_{3min} = -50x_1 + 49x_2 + 33x_3 + 7x_4 - 45x_5$

Kısıtlar:

$21x_1 + 5x_2 + 30x_3 + 42x_4 + 64x_5 \leq 124$

$52x_1 - 4x_2 + 77x_3 + 4x_4 + 14x_5 \leq 62$

$47x_1 + 7x_2 + 51x_3 + 31x_4 + 44x_5 \leq 142$

Doğal Kısıtlar:

$x_{j\geq 0}$, j=1,2,..,5

Problem 4:

$Z_{4min} = -4x_1 - x_2 + 2x_3$

Kısıtlar:

$x_1 - x_2 \leq -2$

$x_1 + 2x_2 + x_3 \leq 8$

$-2x_1 - 6x_2 + 3x_3 \leq -3$

Doğal Kısıtlar:

$x_{j\geq 0}$, j=1,2,3

Problem 5:

$Z_{5min} = 3x_1 - 2x_2 + x_3$

Kısıtlar:

$x_1 + x_2 + x_3 \leq 1$

$-3x_1 + 3x_2 - x_3 \leq 2$

Doğal Kısıtlar:

$x_{j\geq 0}$, j=1,2,3

Problem 6:

$Z_{6min} = -5x_1 - 4x_2 - 6x_3$

Kısıtlar:

$x_1 - x_2 + x_3 \leq 20$

$3x_1 + 2x_2 + 4x_3 \leq 42$

$3x_1 + 2x_2 \leq 30$

Doğal Kısıtlar:

$x_{j\geq 0}$, j=1,2,3

Problem 7:

$Z_{7min} = -x_1 - 3x_2 + x_3$

Kısıtlar:

$-x_1 + x_2 + 2x_3 \leq 5$

$2x_1 + x_2 - 3x_3 \leq 4$

$4x_1 + 3x_2 \leq 4$

Doğal Kısıtlar:

$x_{j\geq 0}$, j=1,2,3

Problem 8:

$Z_{8min} = 7x_1 + 6x_2 + 4x_3 - 8x_4$

Kısıtlar:

$-x_1 + 3x_2 + 2x_3 + x_4 \leq 2$

$2x_1 + 4x_3 - 3x_4 \leq 2$

$4x_1 + 3x_2 - x_3 \leq 1$

Doğal Kısıtlar:

$x_{j\geq 0}$, j=1,2,..,4

Problem 9:

$Z_{9min} = -x_1 + 3x_2 + 2x_3 - 2x_4$

Kısıtlar:

$2x_1 -2x_2 +x_3 +3x_4 \leq 6$

$-x_2 +3x_3 +2x_4 \leq 5$

$5x_1 +6x_2 -x_3 +2x_4 \leq 7$

Doğal Kısıtlar:

$x_{j\geq 0}$, j=1,2,..,4

Problem 10:

$Z_{10min} = x_1 +3x_2 +5x_3 +4x_4$

Kısıtlar:

$2x_1 +4x_2 +7x_3 -x_4 \leq 8$

$-3x_1 +4x_2 +3x_3 \leq 18$

$-3x_1 -7x_2 -2x_3 +2x_4 \leq -12$

Doğal Kısıtlar:

$x_{j\geq 0}$, j=1,2,..,4

Problem 11:

$Z_{11min} = -x_1 + 2x_2 + 2x_3 + 4x_4 + 3x_5 + 5x_6 + 3x_7$

Kısıtlar:

$2x_1 + 7x_2 + 5x_3 + x_4 + 3x_6 + 8x_7 \leq 5$

$-3x_1 + 4x_2 + 6x_3 + 2x_4 + 3x_5 - x_6 + 7x_7 \leq 17$

$3x_1 + 4x_2 - x_3 + 9x_4 + 6x_5 + x_6 + 5x_7 \leq 15$

Doğal Kısıtlar:

$x_{j\geq 0}$, j=1,2,..,7

Problem 12:

$Z_{12min} = 2x_1 + x_2 + 3x_3 + 7x_4 + 3x_5 + 5x_6 + 6x_7 + 2x_8 + 4x_9 + x_{10}$

Kısıtlar:

$2x_1 + 4x_2 - 5x_3 + x_4 + 3x_5 + 6x_6 + 5x_7 + 4x_8 - 9x_9 + 7x_{10} \leq 64$

$-5x_1 - 7x_2 + 6x_3 - 8x_4 - 2x_5 + 4x_6 + 3x_7 + x_8 - 5x_9 + 3x_{10} \leq -58$

Doğal Kısıtlar:

$x_{j\geq 0}$, j=1,2,..,10

Problem 13:

$Z_{13min} = 4x_1 + 5x_2 + 2x_3 + 6x_4 + 4x_5 + x_6 + 2x_7 + 2x_8 + 4x_9 + 6x_{10}$

Kısıtlar:

$3x_1 - 4x_3 + x_4 + 2x_5 + 2x_6 + 3x_7 + 6x_8 + x_9 + 5x_{10} \leq 45$

$5x_2 + x_3 + 2x_4 + 4x_5 - 5x_6 + 6x_7 + 8x_8 - 9x_9 + 4x_{10} \leq 15$

$-2x_1 + 4x_2 - 8x_3 - 4x_5 - 6x_6 + 5x_7 + x_8 + 2x_9 - 5x_{10} \leq -34$

Doğal Kısıtlar:

$x_{j\geq 0}$, j=1,2,..,10

Problem 14:

$Z_{14min} = 2x_1 + 4x_2 + 3x_3 + x_4 + 7x_5 + 5x_6 + 9x_7 + 5x_8 + 4x_9 + 6x_{10}$

Kısıtlar:

$-4x_1 - 5x_2 + 2x_3 - x_4 + 6x_5 - 5x_6 + 7x_7 - 11x_8 + 2x_9 + 5x_{10} \leq -38$

$2x_2 + 5x_3 + 4x_4 - 6x_5 - 5x_6 + 4x_7 - 8x_8 + 5x_9 + 7x_{10} \leq 25$

$x_1 + 5x_2 + 5x_3 + 4x_4 + 6x_6 + 3x_7 - 8x_8 + x_9 + 2x_{10} \leq 17$

$4x_1 + 5x_2 - 5x_3 + 8x_5 - 2x_6 - 4x_7 + x_8 + 3x_9 + 4x_{10} \leq 45$

Doğal Kısıtlar:

$x_{j\geq 0}$, j=1,2,..,10

Problem 15:

$Z_{15min} = x_1 + 4x_2 + 2x_3 + 8x_4 + 4x_5 - 6x_6 + x_7 + 7x_8 - 2x_9 + 6x_{10} + 7x_{11} - x_{12} + 6x_{13} + 8x_{14} + 9x_{15}$

Kısıtlar:

$5x_1 + 4x_2 + 6x_3 + 8x_4 + 5x_5 + 2x_6 + 4x_8 + 2x_9 + 4x_{10} + 2x_{11} + 6x_{12} + 5x_{13} + 8x_{14} + 9x_{15} \leq 64$

Doğal Kısıtlar:

$x_{j\geq 0}$, j=1,2,..,15

Problem 16:

$Z_{16min} = 2x_1 + 5x_2 + 7x_3 + 3x_4 + 6x_5 + 4x_6 + 5x_7 - 7x_8 + 7x_9 + 2x_{10} - 2x_{11} + 4x_{12} - 5x_{13} + 7x_{14} + 3x_{15}$

Kısıtlar:

$-3x_1 + 5x_2 + 5x_4 + 6x_5 + 4x_6 + 8x_7 + 5x_8 + 2x_9 + 4x_{10} + 6x_{11} - 2x_{12} + 3x_{13} + 4x_{14} + 5x_{15} \leq 27$

$4x_1 + 5x_2 + 6x_3 - 2x_4 + 5x_6 + 6x_7 + 5x_9 - 9x_{10} + 2x_{11} + 3x_{12} + 5x_{13} - 7x_{14} + 5x_{15} \leq 12$

Doğal Kısıtlar:

$x_{j\geq 0}$, j=1,2,..,15

Problem 17:

$Z_{17min} = 2x_1 + 3x_2 + 4x_3 + x_4 + x_5 + 2x_6 - 5x_7 + 3x_8 + 4x_9 + 4x_{10} - 2x_{11} + x_{12} + x_{13} + 5x_{14} + 4x_{15}$

Kısıtlar:

$3x_1 + x_2 + 2x_3 + 2x_4 + x_5 + 2x_8 + 4x_9 + 3x_{10} + x_{11} + 2x_{12} + 2x_{13} + x_{14} + 3x_{15} \leq 10$

$3x_2 + 2x_3 + x_4 + 5x_5 + 3x_6 + 3x_7 + 2x_8 + 4x_9 + x_{10} + 2x_{11} + x_{12} + 4x_{13} + 2x_{14} + 3x_{15} \leq 16$

Doğal Kısıtlar:

$x_{j \geq 0}$, j=1,2,..,15

Problem 18:

$Z_{18min} = 9x_1 + 7x_2 + 5x_3 - 6x_4 + 8x_5 + 7x_6 - 7x_7 + 4x_8 - 3x_9 + 4x_{10} + 6x_{11} + x_{12} - 5x_{13} + 3x_{14} + 8x_{15} - 2x_{16} + 4x_{17}$

Kısıtlar:

$5x_1 + 6x_2 + 7x_3 + 8x_4 + 6x_5 + 4x_6 + 5x_7 - x_8 + 2x_9 + 3x_{10} + 5x_{11} + 4x_{12} + 6x_{13} + 4x_{14} + 8x_{15} + 9x_{16} + 5x_{17} \leq 270$

$2x_1 + 4x_2 + 5x_3 + 6x_4 + 5x_5 + 7x_6 + 4x_7 + 4x_8 + 5x_9 + x_{11} - 5x_{12} + 6x_{13} + 5x_{14} + 5x_{15} + 4x_{16} + x_{17} \leq 180$

$-5x_1 - 6x_2 - 4x_3 + 2x_4 - 4x_5 - x_6 + 6x_7 - 7x_8 - 5x_9 - 9x_{10} - 5x_{11} + 5x_{12} - 4x_{13} - 3x_{14} - 2x_{15} - x_{16} - 8x_{17} \leq -120$

Doğal Kısıtlar:

$x_{j\geq 0}$, j=1,2,..,17

Problem 19:

$Z_{19min} = 5x_1 + 7x_2 + 3x_3 + x_4 + 5x_5 + 4x_6 - 2x_7 + 9x_8 + x_9 + x_{10} + 4x_{11} + 6x_{12} + 3x_{13} - 8x_{14} + 7x_{15} + x_{16} + 2x_{17}$

Kısıtlar:

$3x_1 + 5x_2 + 5x_3 + 4x_4 + 2x_5 + x_6 + 5x_7 + 6x_8 + 2x_9 + 6x_{10} + 5x_{11} + 8x_{12} + 8x_{13} + 6x_{14} + 2x_{15} + 2x_{16} + x_{17} \leq 59$

$2x_1 + 4x_4 + 5x_5 + 6x_6 + 8x_7 + 4x_8 + 4x_9 + 2x_{10} + 5x_{11} + 5x_{12} + 4x_{13} + 3x_{14} + 6x_{15} + 5x_{16} + 9x_{17} \leq 75$

$4x_1 + 5x_2 + 5x_3 + 7x_4 + 8x_5 + 6x_6 + 2x_7 + 5x_8 + 4x_9 + 3x_{10} + 2x_{11} + 2x_{12} + 5x_{13} + 3x_{14} + 5x_{15} + 6x_{16} + 5x_{17} \leq 80$

$6x_1 - 6x_2 - 4x_3 + 2x_4 - 3x_5 - 5x_6 - 5x_7 - 8x_8 - x_9 - 2x_{10} - 4x_{12} - 6x_{13} - 3x_{14} - 3x_{15} - 7x_{16} - 2x_{17} \leq -80$

Doğal Kısıtlar:

$x_{j\geq 0}$, j=1,2,..,17

Problem 20:

$Z_{20min} = -x_1 + 4x_2 + 5x_3 - 9x_4 + 7x_5 + 2x_6 - 8x_7 + 3x_8 - 2x_9 + 2x_{10} + 4x_{11} + 8x_{12} + 7x_{13} + 6x_{14} + 3x_{15} - 6x_{16} + 4x_{17} - 7x_{18} + 2x_{19} + 7x_{20}$

Kısıtlar:

$-24x_1 -2x_2 -5x_3 +6x_4 -12x_5 +3x_6 +5x_7 +9x_8 +8x_9 -3x_{10} +6x_{11} +5x_{12} +4x_{13} +63x_{14} -5x_{15} +5x_{16} +2x_{17} +3x_{18} +x_{19} -7x_{20} \leq -120$

$4x_2 +6x_3 +50x_4 -4x_6 +9x_7 +13x_8 -4x_9 +6x_{10} +5x_{11} +2x_{12} +x_{13} +5x_{14} +5x_{15} +7x_{16} +9x_{17} +5x_{18} +6x_{19} +2x_{20} \leq 35$

$4x_1 +2x_2 +3x_3 +2x_4 -x_5 +5x_6 +6x_7 +7x_8 +6x_9 +2x_{10} -3x_{11} +9x_{12} +4x_{14} +6x_{15} +5x_{16} +4x_{17} +3x_{18} +2x_{19} +x_{20} \leq 42$

$4x_1 +2x_2 +x_3 +5x_4 +6x_5 -5x_6 +3x_7 +2x_8 +5x_9 +6x_{10} +8x_{11} -7x_{12} -9x_{13} +5x_{14} +6x_{15} +2x_{16} +3x_{17} +2x_{18} +3x_{19} +7x_{20} \leq -150$

Doğal Kısıtlar:

$x_{j\geq 0}$, j=1,2,..,20

Matlab Programı üzerinde linprog komutu kullanılarak 20 problem çözdürülmüş ve aşağıdaki sonuçlar alınmıştır.

Çizelge 4.3: Linprog komutu ile alınan sonuçlar ve süreleri

Problem No	SÜRE	ÇÖZÜM
1	0,010095	13,125
2	0,010352	-70
3	0,009948	-113,108
4	0,008627	-8,66667
5	0,011402	-1,33333
6	0,009771	-78
7	0,011831	-4
8	0,011148	-16,25
9	0,011924	-4
10	0,009821	4
11	0,010694	-2,5
12	0,017658	8,285714
13	0,017546	5,666667
14	0,011111	17,27273
15	0,010593	-192
16	0,010441	-44,4
17	0,010135	-26,6667
18	0,01454	-194,8
19	0,012873	-47,8056
20	0,015616	122,8784

4.3. DOĞRUSAL PROGRAMLAMA PROBLEMLERİ İÇİN META SEZGİSEL YÖNTEMLERİN DÜZENLENMESİ

Bölüm 3'te anlatılan optimizasyon algoritmalarının doğrusal programlama problemlerini çözmedeki başarılarını test edebilmek için 20 farklı problem belirli koşullar altına çözdürülmektedir. Koşullar Çizelge 4.4'de sıralanmıştır.

Çizelge 4.4: Algoritmalarda kullanılan sürü ve iterasyon sayıları

Algoritma	**Sürü Sayısı**	**İterasyon**
Genetik Algoritma (Matlab)	Popülasyon Matlab default (20)	jenerasyon Matlab default (100)
Karınca Kolonisi Optimizasyonu	Karınca Sayısı 400	İterasyon 400
Parçacık Sürü Optimizasyonu	Kuş Sayısı 400	İterasyon 400
Benzetilmiş Tavlama (Matlab)	Başlangıç Sıcaklığı Matlab default (100)	İterasyon Matlab default (500*değişken)
Tabu Arama Algoritması	Aday Çözüm Sayısı 400	İterasyon 400

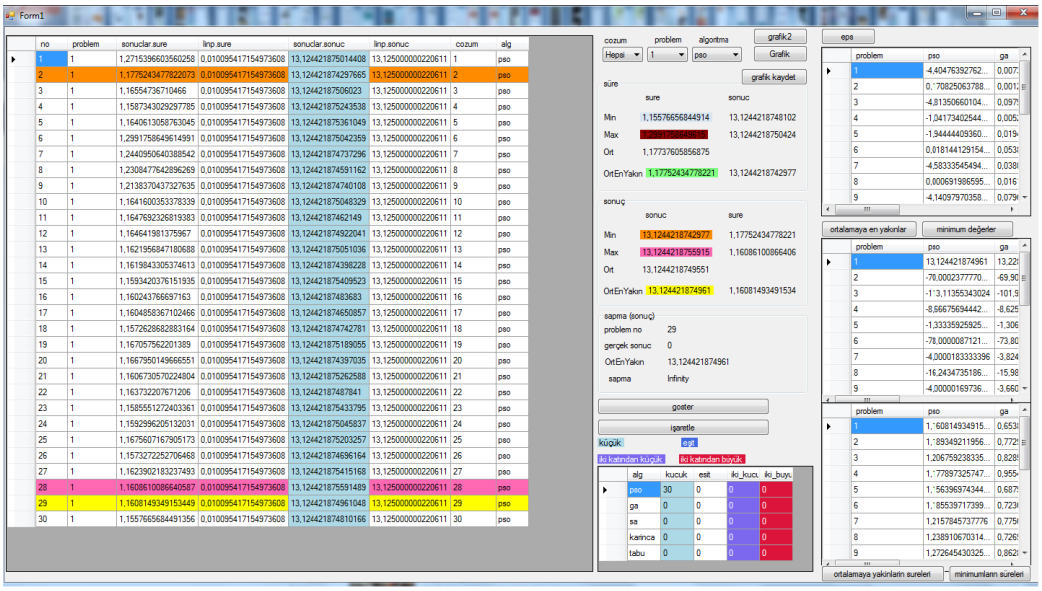
Karınca Koloni Optimizasyon Algoritması, Parçacık Sürü Optimizasyon Algoritması, Tabu Arama Algoritması için başlangıç noktaları belirlenirken $x_{0i}=\alpha_p$ * rnd (i=1,2,.. değişken sayısı) fonksiyonunda kullanılan α değerleri her problem için Çizelge 4.5'te verilmiştir.

Çizelge 4.5: Başlangıç noktalarını belirlemede kullanılan α değerleri

Problem No	**α**
1	7
2	10
3	20,2
4	8
5	1
6	15
7	2,5
8	3
9	2,5
10	4
11	2,5
12	64
13	45
14	45
15	32
16	8
17	8
18	90
19	19,6
20	35

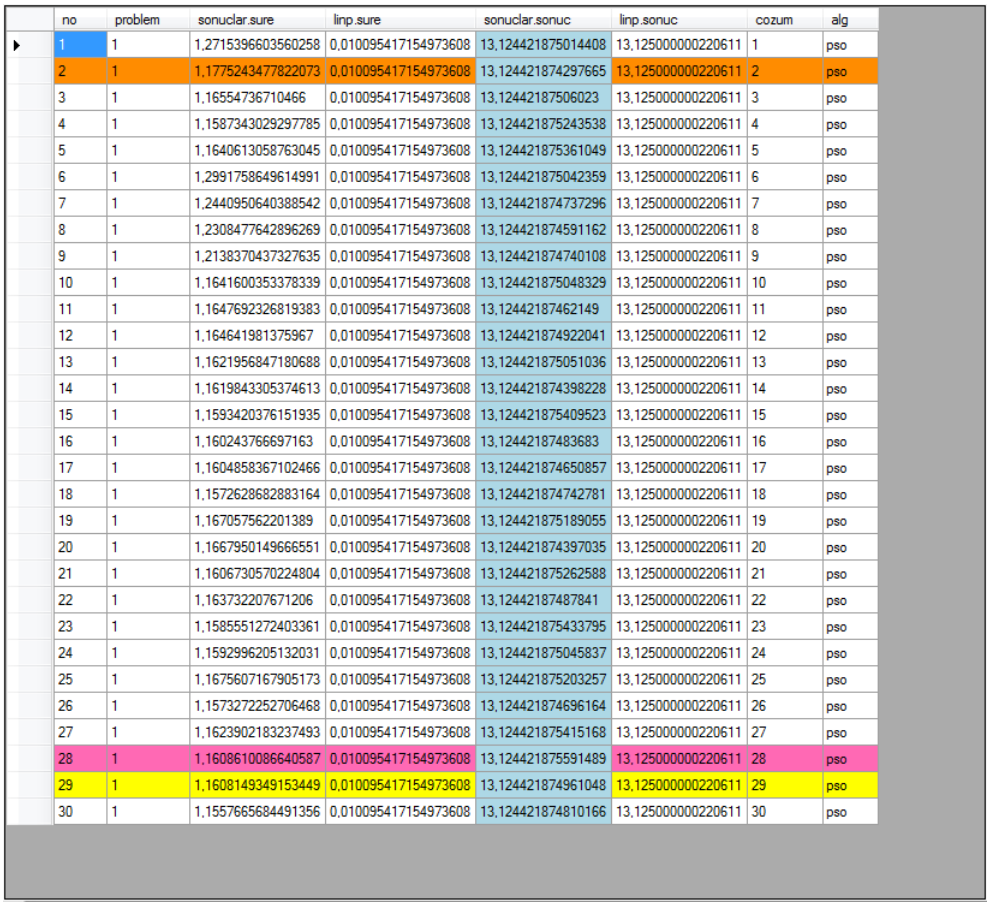
4.4. VERİ ANALİZİ İÇİN KULLANILAN PROGRAM

Doğrusal programlama problemlerinin çözümünden sonra verilerin analiz edilebilmesi için VB.Net platformu kullanılarak bir arayüz hazırlanmıştır. Veritabanına kaydedilen veriler çözüm, problem veya algoritmaya göre süzdürülerek incelenebilmektedir. Çizelge üzerinde çözümler süreleriyle birlikte linprog komutunun çözümleri de görülecek şekilde görüntülenebilmektedir. Yapılan süzdürme işleminden sonra süre veya sonuç olarak minimum, maksimum ve ortalamaya en yakın değerler çizelge üzerinde işaretlenmektedir. Beş algoritma için 20 problemin 30 adet çözümünün bağıl hata değerleri bir tabloya aktarılabilmektedir. Aynı şekilde beş algoritma için 20 problemin 30 adet çözümünün ortalaması alınarak ortalamaya en yakın olan çözüm veya minimum çözüm ve bu çözümün süresi ayrı tablolara aktarılabilmektedir.



Şekil 4.11: Veri analizi için kullanılan programın ekran görüntüsü

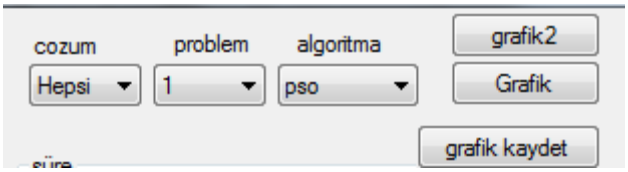
Program üzerindeki panellerin ayrıntılı görüntüleri aşağıdaki şekillerde verilmiştir.



no	problem	sonuclar.sure	linp.sure	sonuclar.sonuc	linp.sonuc	cozum	alg
1	1	1,2715396603560258	0,010095417154973608	13,124421875014408	13,125000000220611	1	pso
2	1	1,1775243477822073	0,010095417154973608	13,124421874297665	13,125000000220611	2	pso
3	1	1,16554736710466	0,010095417154973608	13,12442187506023	13,125000000220611	3	pso
4	1	1,1587343029297785	0,010095417154973608	13,124421875243538	13,125000000220611	4	pso
5	1	1,1640613058763045	0,010095417154973608	13,124421875361049	13,125000000220611	5	pso
6	1	1,2991758649614991	0,010095417154973608	13,124421875042359	13,125000000220611	6	pso
7	1	1,2440950640388542	0,010095417154973608	13,124421874737296	13,125000000220611	7	pso
8	1	1,2308477642896269	0,010095417154973608	13,124421874591162	13,125000000220611	8	pso
9	1	1,2138370437327635	0,010095417154973608	13,124421874740108	13,125000000220611	9	pso
10	1	1,1641600353378339	0,010095417154973608	13,124421875048329	13,125000000220611	10	pso
11	1	1,1647692326819383	0,010095417154973608	13,12442187462149	13,125000000220611	11	pso
12	1	1,164641981375967	0,010095417154973608	13,124421874922041	13,125000000220611	12	pso
13	1	1,1621956847180688	0,010095417154973608	13,124421875051036	13,125000000220611	13	pso
14	1	1,1619843305374613	0,010095417154973608	13,124421874398228	13,125000000220611	14	pso
15	1	1,1593420376151935	0,010095417154973608	13,124421875409523	13,125000000220611	15	pso
16	1	1,160243766697163	0,010095417154973608	13,12442187483683	13,125000000220611	16	pso
17	1	1,1604858367102466	0,010095417154973608	13,124421874650857	13,125000000220611	17	pso
18	1	1,1572628682883164	0,010095417154973608	13,124421874742781	13,125000000220611	18	pso
19	1	1,167057562201389	0,010095417154973608	13,124421875189055	13,125000000220611	19	pso
20	1	1,1667950149666551	0,010095417154973608	13,124421874397035	13,125000000220611	20	pso
21	1	1,1606730570224804	0,010095417154973608	13,124421875262588	13,125000000220611	21	pso
22	1	1,163732207671206	0,010095417154973608	13,12442187487841	13,125000000220611	22	pso
23	1	1,1585551272403361	0,010095417154973608	13,124421875433795	13,125000000220611	23	pso
24	1	1,1592996205132031	0,010095417154973608	13,124421875045837	13,125000000220611	24	pso
25	1	1,1675607167905173	0,010095417154973608	13,124421875203257	13,125000000220611	25	pso
26	1	1,1573272252706468	0,010095417154973608	13,124421874696164	13,125000000220611	26	pso
27	1	1,1623902183237493	0,010095417154973608	13,124421875415168	13,125000000220611	27	pso
28	1	1,1608610086640587	0,010095417154973608	13,124421875591489	13,125000000220611	28	pso
29	1	1,1608149349153449	0,010095417154973608	13,124421874961048	13,125000000220611	29	pso
30	1	1,1557665684491356	0,010095417154973608	13,124421874810166	13,125000000220611	30	pso

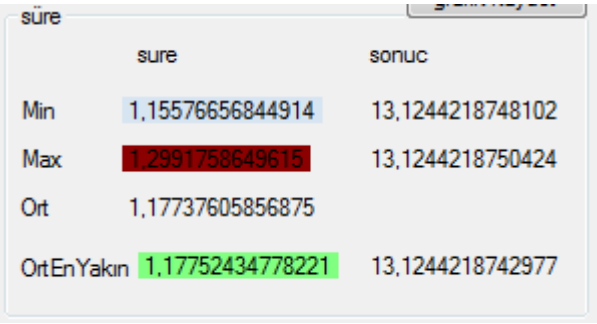
Şekil 4.12: Veri görüntüleme paneli

Şekil 4.13'de gerçekleştirilen seçim kriterlerine göre Şekil 4.12'de görülmekte olan verilen süzdürülmüştür. Tablo üzerinde görülen renklendirmelerin anlamları Şekil 4.18'te verilmiştir.



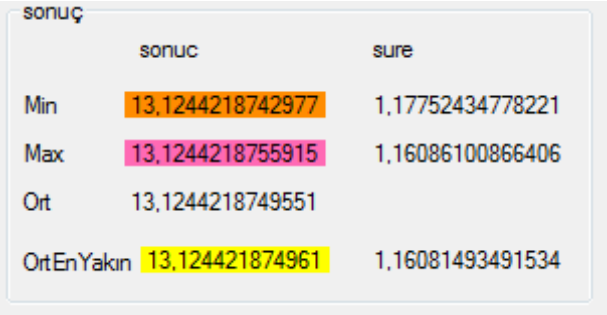
Şekil 4.13: Çözüm, Problem, Algoritma seçimi ve grafik dökümleri

Veri setinin süzdürülmesi çözüm, problem ve algoritma kriterlerine göre yapılmaktadır. Süzdürülen verilerin grafikleri Şekil 4.13'de grafik butonuna basılarak oluşturulmaktadır.



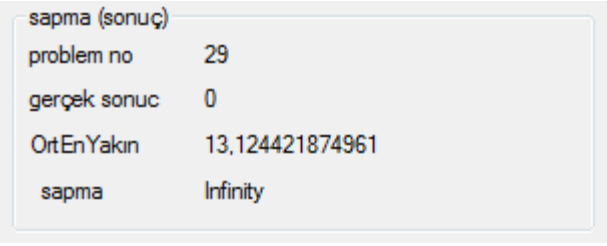
Şekil 4.14: Seçim kriterlerine göre min, max, ortalama ve ortalamaya en yakın süre değerleri

Seçim kriterlerine göre min, max, ortalama ve ortalamaya en yakın süre değerleri ve bu değerlere ulaşan çözümlerin sonuçları Şekil 4.14'da yazdırılmıştır. Renklendirilmiş etiketlere tıklanarak değerler tablo üzerinde işaretlenebilmektedir.



Şekil 4.15: Seçim kriterlerine göre min, max, ortalama ve ortalamaya en yakın sonuç değerleri

Seçim kriterlerine göre min, max, ortalama ve ortalamaya en yakın sonuç değerleri ve bu değerlere ulaşan çözümlerin süreleri Şekil 4.15'te yazdırılmıştır. Renklendirilmiş etiketlere tıklanarak değerler tablo üzerinde işaretlenebilmektedir.



Şekil 4.16: Ortalamaya en yakın sonucun çözüm numarası

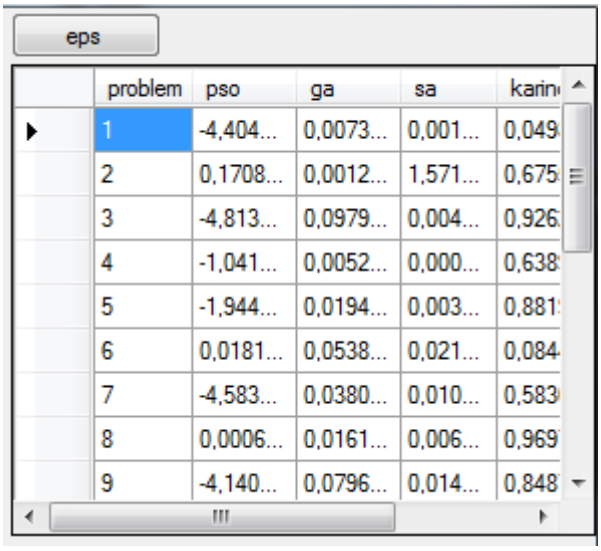
Seçim kriterlerine göre ortalamaya en yakın sonuç ve o sonuca ulaşan çözüm numarası Şekil 4.16'da verilmiştir.

Şekil 4.17: Süre, sonuç ve sonuç bazında ortalamaya en yakın değerlerin gösterimi

Süre, sonuç ve sonuç bazında ortalamaya en yakın değerlerin gösterimi Şekil 4.17'de göster butonuna basılarak gerçekleştirilmiştir.

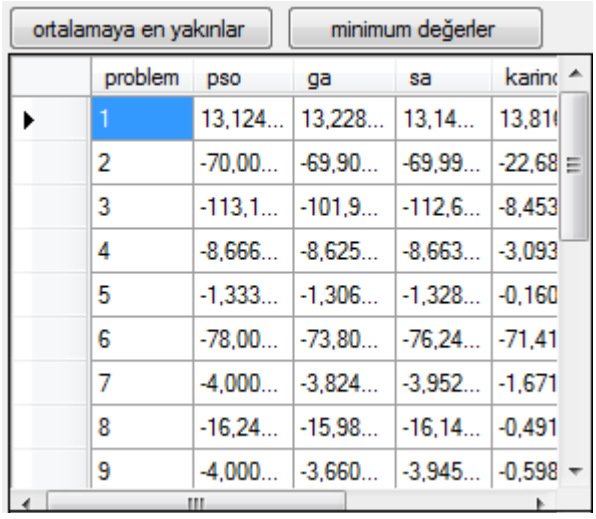
Şekil 4.18: Seçim kriterlerine göre algoritmaların linprog komutu ile karşılaştırılma tablosu

Seçim kriterlerine göre algoritmaların linprog komutu ile karşılaştırma tablosu Şekil 4.18'te verilmiştir. Tablo üzerinde küçük değeri linprog sonucundan küçük sonuçları, eşit değeri linprog sonucuna eşit olan sonuçları, iki küçük değeri linprog sonucunun iki katından küçük, iki büyük değeri ise linprog sonucunun iki katından da büyük olduğu sonuçları temsil etmektedir. Bu değerlerin adedi Şekil 4.18'te verilmekte ve Şekil 4.12'deki tablo üzerinde işaretlenerek sonuçların durumu görülebilmektedir.



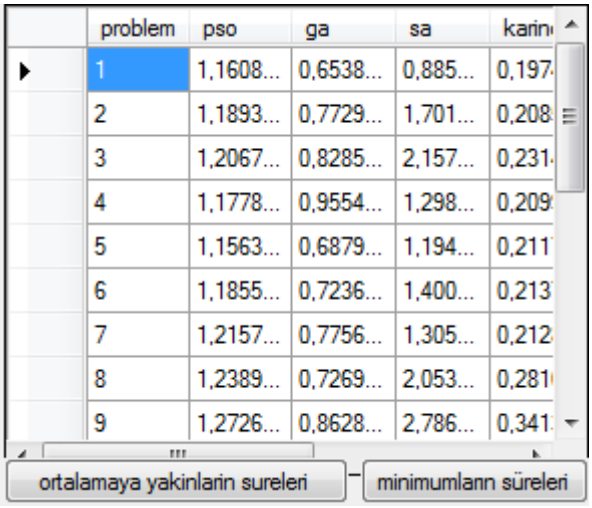
Şekil 4.19: Bağıl hata döküm tablosu

Şekil 4.19'te görülmekte olan eps butonuna basıldığında tüm problemler ve algoritmalar için gerçekleştirilen çözümler içersinde bağıl hata değerleri hesaplanarak tabloya yazılmaktadır.



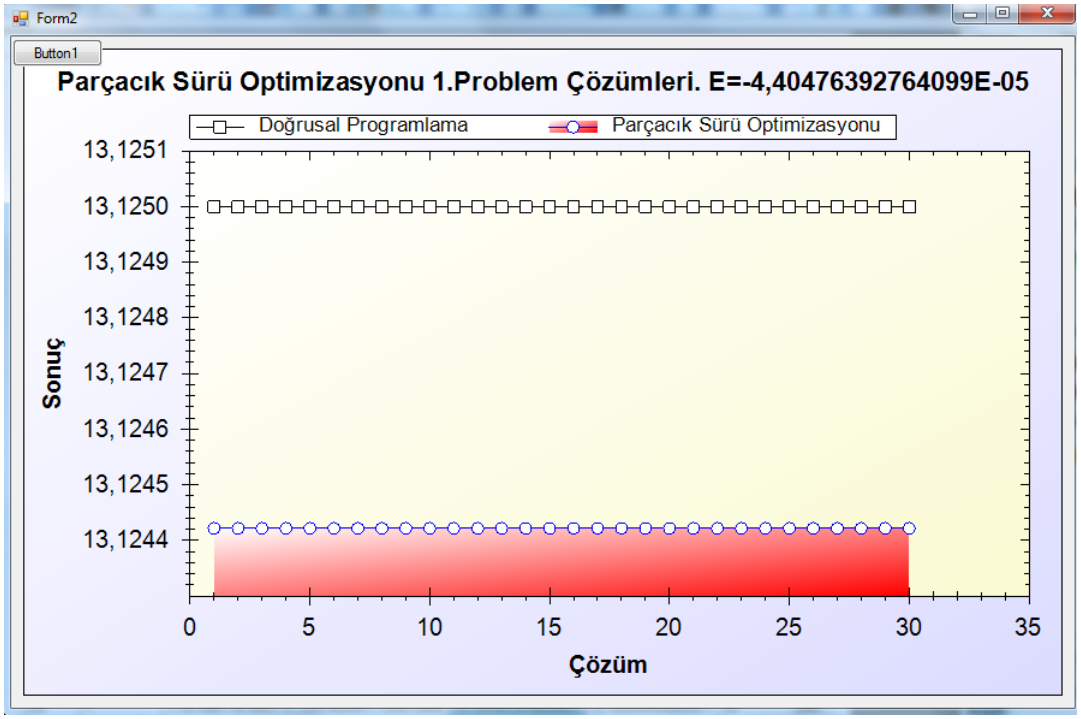
Şekil 4.20: Ortalamaya en yakın veya minimum sonuçlar

Şekil 4.20'te görülen ortalamaya en yakınlar butonuna basılarak tüm problem ve algoritmalar için gerçekleştirilen çözümler içersine ortalamaya en yakın olan çözümler, minimum değerler butonuna basılarak tüm problemler ve algoritmalar için gerçekleştirilen çözümler içersinde minimum değere sahip olan çözümler bulunarak tabloya yazılmaktadır.



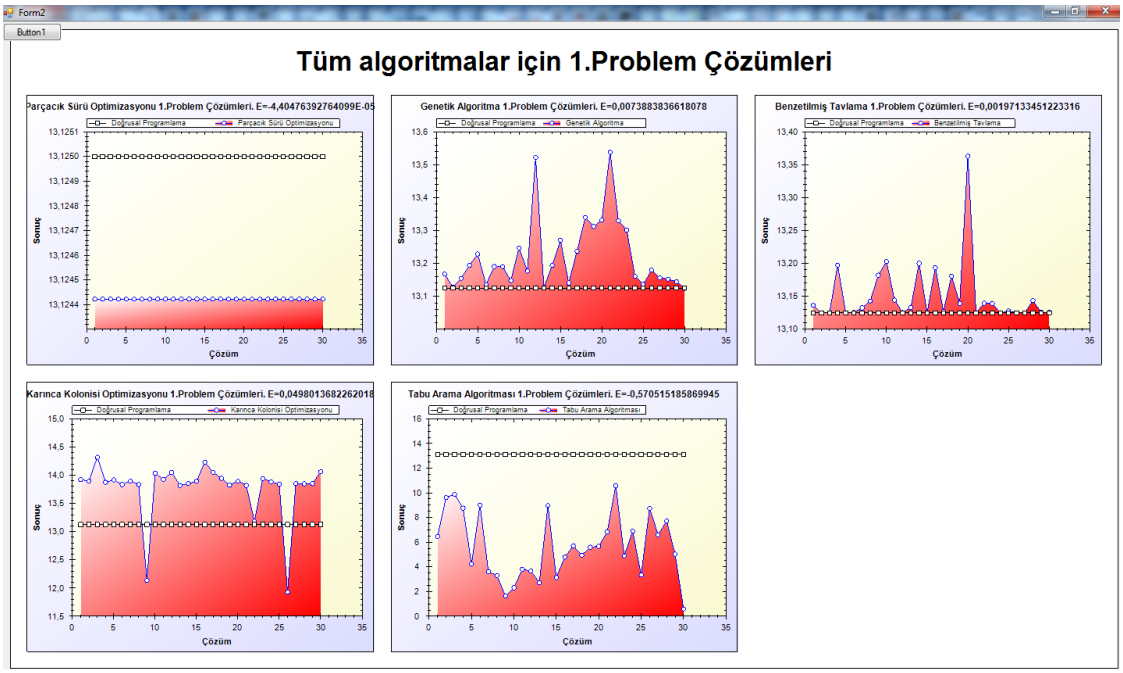
Şekil 4.21: Ortalamaya en yakın veya minimum sonuçların çözüm süreleri

Şekil 4.21'da görülen ortalamaya en yakınların süreleri butonuna basılarak tüm problem ve algoritmalar için gerçekleştirilen çözümler içersine ortalamaya en yakın olan çözümlerin çözüm süreleri, minimumların süreleri butonuna basılarak tüm problemler ve algoritmalar için gerçekleştirilen çözümler içersinde minimum değere sahip olan çözümlerin çözüm süreleri bulunarak tabloya yazılmaktadır.



Şekil 4.22: Analiz programı üzerinde sonuç grafiğinin oluşturulması

Analiz programı üzerinde grafiklerin oluşturulması için Zedgraph modülü kullanılmıştır. Bu modül kullanılarak her problemin 30 adet çözümü linprog komutunun çözümü ile karşılaştırmalı olarak grafik ekranda görüntülenmiştir.



Şekil 4.23: Analiz program üzerinde çoklu sonuç grafiği incelenmesi

Analiz programı üzerinde çoklu tablo gösterimi aktifleştirilerek her algoritmanın tek problem için çözüm grafikleri aynı ekranda görüntülenmiştir.

5. BULGULAR

Rastgele üretilmiş 20 problem için; 4. Bölümde verilen kriterler ile çalıştırılan GA (Genetik Algoritma), KKO (Karınca Koloni Optimizasyonu), PSO (Parçacık Sürü Optimizasyonu), BT (Benzetilmiş Tavlama), TA (Tabu Arama) algoritmaları kullanılarak gerçekleştirilen çözümlerin sonuç bilgileri aşağıda verilmiştir.

Problem çözümlerinde elde edilen bağıl hata değerleri Çizelge 5.1'de düzenlenmiş şekilde verilmektedir. Bağıl hata değerleri hesaplanırken Denklem (5.1) kullanılmıştır. Denklemde α değeri problemin linprog komutu ile çözümünden elde edilen sonucu, β değeri problemin belirtilen algoritma ile i=(1,2, .. , n) için çözümünden elde edilen sonucu ifade etmektedir.

$$\varepsilon_{bağıl} = \frac{\left|\alpha - \frac{\sum_{i=1}^{n} \beta}{n}\right|}{|\alpha|} \tag{5.1}$$

Problem çözümlerinde elde edilen bağıl hata değerleri Çizelge 5.1'de görülmektedir.

Çizelge 5.1: Problem çözümlerinden elde edilen bağıl hata değerleri

Problem	GA (Matlab)	KKO	PSO	BT (Matlab)	TA
1	0,011734311	0,005695721	0,31725806	0,002217853	0,025022514
2	0,002267826	0,190092122	0,562094211	2,61286E-05	0,054526755
3	0,127593806	3,487382043	4,83628E-05	16082,35887	3,814154367
4	0,005969332	0,265655469	0,201284952	0,000199852	0,506272944
5	0,017485668	0,077301563	0,44937661	0,006900814	0,019960332
6	0,04545559	0,03215456	0,014517398	0,024427835	0,67304834
7	0,019058599	0,097276046	0,191912271	0,010315753	0,229050217
8	0,031436206	0,610182277	0,483893473	0,005978606	0,282438532
9	0,057043167	0,277904939	1,038962326	0,011255633	0,079598738
10	0,227535209	1,562941994	1,280021827	349,973298	0,149028852
11	0,75972036	3,575286039	0,551232653	0,213573082	1,709700246
12	2,910145961	68,51452942	467,2727161	1,434044407	39,65175298
13	2,20415905	82,93932322	15,13208259	2,451175788	39,18881069
14	0,39891162	32,85088255	6,446601638	0,319353749	16,73073241
15	0,751230247	2,104296148	0,017822147	780919445,4	1,486469293
16	0,463511644	2,158857023	0,206703946	0,552123992	1,289371641
17	0,417627611	1,384550371	0,059412153	0,179306058	1,184695216
18	0,589632197	6,010262954	0,350582627	0,383304298	1,035230158
19	1,778257499	3,150896773	0,486138832	6,413006968	2,085802422
20	0,161373067	0,767351071	3,24203774	345038,5334	1,191378111

Her algoritma için gerçekleştirilen tüm çözümlerin bağıl hata oranlarının ortalaması alınarak bir sıralama yapılmak istenirse iyiden kötüye doğru GA, TA, KKO, PSO ve BT algoritmaları şeklinde ifade edilebilir.

Bağıl hata oranlarına bakıldığında; GA algoritması 12, 13, 19. Problemler olmak üzere toplam 3 problemde olağan sonuçtan hayli uzaklaşmış durumdadır. KKO algoritması ise 3,10-19. Problemler olmak üzere toplam 11 problemde uzaksamaktadır. PSO algoritması 9, 10, 12-14, 20. problemler olmak üzere toplam 6 problemde olağan sonuçtan uzaklaşmıştır. BT algoritması 3, 10, 12, 13, 15, 19, 20. problemler olmak üzere toplam 7 problemde 1'in üzerinde bağıl hata değeri elde etmiştir. TA algoritması ise 3, 11-20. problemler olmak üzere toplam 11 problemde olağan sonuçtan uzaklaşmıştır. Bağıl hata oranlarının 1'in üzerinde olma durumu göz önüne alınarak sıralama yapıldığında; iyiden kötüye doğru GA, PSO, BT, TA ve KKO algoritmaları şeklinde ifade edilebilir.

Yukarıdaki veriler göz önünde bulundurulduğunda GA ve PSO algoritmalarının benzer koşullar altında diğer algoritmalara göre daha iyi sonuçlara ulaştığı söylenebilir.

Her bir problemde gerçekleştirilen 30 çözüm içersinde ortalamaya en yakın olan sonuç değerleri her bir algoritma için Çizelge 5.2'de listelenmiştir.

Çizelgede bulunan linprog sütunu, matlab programında doğrusal programlama (linprog) çözümlerini içermektedir.

Çizelge 5.2: Her algoritma için ortalamaya en yakın çözümler

Problem	**linprog**	**GA (Matlab)**	**KKO**	**PSO**	**BT (Matlab)**	**TA**
1	13,125	13,26911508	13,195727	17,2034009	13,15861581	13,0741015
2	-70	-69,8428653	-56,3773776	-29,2680292	-69,9986952	-66,21565
3	-113,108	-101,228255	242,655472	-113,113577	-108,732319	-550,10689
4	-8,66667	-8,62009267	-6,3425287	-7,01134694	-8,66454832	-13,2470068
5	-1,33333	-1,31300546	-1,2274134	-0,7342028	-1,32484435	-1,30770754
6	-78	-74,5148916	-75,547698	-78,0000087	-76,1419705	-128,338550
7	-4	-3,92171250	-3,61155095	-2,99539012	-3,95785546	-5,06583702
8	-16,25	-15,7758173	-6,38153584	-9,23479229	-16,1556009	-20,8462958
9	-4	-3,77625276	-2,85455072	0,16109786	-3,95296534	-3,68257372
10	4	4,916188525	10,2765705	11,6361736	4,633761037	4,60308933
11	-2,5	-0,62008029	6,53215962	-2,5001712	-1,97572213	1,87253657
12	8,285714	32,05421326	570,526235	1757,67670	20,62167547	341,826950
13	5,666667	18,1875305	476,2265873	5,887832701	19,56690522	228,0005627
14	17,27273	24,16533303	565,1646796	22,48456098	22,72477937	306,1534984
15	-192	-46,2189291	244,3348649	-187,820328	-98,3265160	92,25763683
16	-44,4	-23,9192258	52,0775957	-34,6450573	-20,8161667	11,84071718
17	-26,6667	-15,4877989	11,30518147	-25,1118105	-22,1148562	4,379409244
18	-194,8	-80,2983151	986,4658836	-124,375911	-121,012561	7,459162971
19	-47,8056	37,27985624	104,2344856	-25,0910541	30,65983351	52,74763946
20	122,8784	142,5582715	232,6742817	542,2313954	44043975,64	-25,1859052

Gerçekleştirilen 30 çözümün ortalamasına en yakın çözümler algoritmalar bazında karşılaştırıldığında:

GA algoritması sadece 20. problemde diğer algoritmalara nazaran daha iyi bir sonuca ulaşmıştır.

KKO algoritması diğer algoritmalara göre daha iyi bir sonuca hiç ulaşamamıştır.

PSO algoritması 3, 6, 11, 13-19. problemler olmak üzere toplam 10 problemde daha iyi sonuca ulaşmıştır.

BT algoritması 1, 2, 4, 5, 7-9, 12. problemler olmak üzere toplam 8 problemde daha iyi sonuca ulaşmıştır.

TA algoritması sadece 10. problemde diğer algoritmalara nazaran daha iyi bir sonuca ulaşmıştır.

Yukarındaki verilere dayanarak PSO ve BT algoritmalarının benzer koşullar altında diğer algoritmalara nazaran daha iyi sonuçlar ürettiği söylenebilir.

Çizelge 5.3: Her algoritma için ortalamaya en yakın çözümlerin süreleri

	GA (Matlab)	**KKO**	**PSO**	**BT (Matlab)**	**TA** (sn)
1	0,810582043	7,64365612	4,531907534	1,856146055	5,089345775
2	0,817992603	7,757482413	4,530091643	2,083914923	7,945615425
3	0,810759025	7,714829092	4,735765588	10,82668445	8,08668593
4	6,40476227	7,699204972	4,523987237	2,106866232	7,940533417
5	2,558007215	7,703903763	4,472745184	2,173163431	7,883431222
6	0,757925599	7,741184008	4,581357099	1,290850412	7,920766316
7	0,753065184	7,90379655	4,606548469	1,386138969	8,214290468
8	0,961887503	8,11896827	4,701808504	2,682729584	8,145329036
9	0,710202703	7,88126795	4,704237249	2,727278511	8,156099323
10	0,833999209	7,895514977	4,656704498	2,56859467	8,14231669
11	0,818241255	8,045097424	5,044458974	4,679557809	8,349269731
12	0,746415938	8,16617046	4,984040931	9,671813763	8,541770241
13	0,839272093	8,27613314	5,285863477	5,549223219	8,512528037
14	0,874255964	8,36853514	5,14093009	8,21707537	8,779474949
15	0,935587437	8,27631451	5,216684109	27,76506557	8,569653635
16	1,015647523	8,492059592	5,463667192	12,51923652	8,76207443
17	0,911484284	8,470380796	5,486357417	9,250533688	8,744570794
18	1,085218152	8,736733869	5,615007024	42,77370403	8,939415946
19	1,323048649	8,764841781	5,706782275	43,35533172	9,057577559
20	1,335925165	8,833648902	5,952494308	52,98196322	9,20473565
Ortalama Süre	**1,265213991**	**8,124486186**	**4,99707194**	**12,32329361**	**8,249274229**

Her problem için gerçekleştirilen 30 çözüm içinde ortalamaya en yakın olan sonuç değerlerinin üretilmesinde harcanan süreler tüm algoritmalar için Çizelge 5.3'te listelenmiştir.

Çözümlerin süreleri göz önünde bulundurulduğunda BT, TA ve KKO algoritmaları en yavaş sonuca ulaşan algoritmalardır. GA algoritmasının süreleri ise çok iyidir. Süre değerleri incelendiğinde GA ve PSO algoritmaları öne çıkarken ortalamaya en yakın çözüm değerleri bu karşılaştırmaya eklendiğinde PSO algoritması ön plana çıkmaktadır.

Çizelge 5.4: Her algoritma için minimum çözümler

Problem	**linprog**	**GA (Matlab)**	**KKO**	**PSO**	**BT (Matlab)**	**TA**
1	13,125	13,12482849	13,11320358	13,12442187	13,12500014	6,24034358
2	-70	-70,0012025	-64,8094740	-70,0002377	-70,00000049	-69,1063864
3	-113,108	-112,7789883	-123,090653	-113,113655	-113,1077823	-1057,41119
4	-8,66667	-8,670654549	-8,47010788	-19,3927827	-8,666683794	-28,2748591
5	-1,33333	-1,333986514	-1,28833948	-1,33335925	-1,332598233	-1,32402071
6	-78	-77,99595293	-77,1580806	-86,8509166	-77,75442448	-195,898459
7	-4	-4,000827886	-3,92125594	-5,34379716	-3,999986036	-7,78078783
8	-16,25	-16,15259695	-14,362537	-16,2503499	-16,24791084	-23,0616123
9	-4	-4,000662767	-3,65424853	-4,00000174	-3,994131012	-4,09770527
10	4	4,532320993	4,771803246	3,999979318	4,002324651	4,062355923
11	-2,5	-2,392254429	2,230541556	-2,50017125	-2,438548134	0,07422993
12	8,285714	21,71316926	393,9307567	8,285343962	14,52433257	286,8341825
13	5,666667	8,381922464	264,9598677	5,666519629	7,967299184	136,6373218
14	17,27273	19,81420762	43,81533813	17,26795747	19,95652321	162,1678255
15	-192	-162,3780418	-157,486719	-192,150984	-189,2790427	-4,89726054
16	-44,4	-37,74334973	-9,77832641	-42,6914592	-42,49400049	-23,4262792
17	-26,6667	-21,90699202	-37,6879739	-26,6534521	-25,21092177	-8,54819624
18	-194,8	-120,8882872	-161,192738	-186,678253	-186,6524148	-502,974703
19	-47,8056	2,427768909	-41,4893798	-46,6641624	-46,46076099	5,671071704
20	122,8784	131,4322223	-428,8330948	132,0533073	184,5128892	-139,5867332

Her bir problemde gerçekleştirilen 30 çözüm içersinde minimum değere sahip olan sonuç değerleri her bir algoritma için Çizelge 5.4'te listelenmiştir.

Gerçekleştirilen 30 çözüm içinden en küçük olan değerler algoritmalar bazında incelendiğinde:

GA algoritması 6, 20. problemler olmak üzere toplam 2 problemde diğer algoritmalara nazaran daha iyi bir sonuca ulaşmıştır.

KKO algoritması diğer algoritmalara göre daha iyi bir sonuca hiç ulaşamamıştır.

PSO algoritması 5, 8-19. problemler olmak üzere toplam 13 problemde daha iyi sonuca ulaşmıştır.

BT algoritması 1-4, 7. problemler olmak üzere toplam 5 problemde daha iyi sonuca ulaşmıştır.

TA algoritması diğer algoritmalara göre daha iyi bir sonuca hiç ulaşamamıştır.

Çizelge 5.5: Her algoritma için minimum çözümlerin üretilmesinde harcanan süreler

	GA (Matlab)	**KKO**	**PSO**	**BT (Matlab)**	**TA** (sn)
1	0,729920804	7,666913125	4,54171466	1,247813874	5,097131507
2	0,74142169	7,762525661	4,596425408	1,524673224	7,954076174
3	0,837484724	7,729876193	4,76767495	6,546431734	7,95787031
4	0,911972812	7,70255446	4,51917509	2,540407043	7,949340816
5	4,253344918	7,695276271	4,52581556	1,81619207	7,909355384
6	0,712339647	7,718301444	4,488108951	2,666123289	7,921785789
7	0,677439879	7,909604768	4,60158128	1,629630686	8,199554913
8	0,876411923	7,907455391	4,803511551	3,832128468	8,151350071
9	4,470186262	8,00443405	4,787521766	1,52064945	8,138969396
10	0,790046315	7,880157792	4,809277352	8,482799499	8,1365838
11	0,914539778	8,078612054	5,018871222	6,700907507	8,348158842
12	0,821306257	8,176195522	5,189358719	11,96292672	8,555788362
13	1,061473351	8,261856129	5,273171256	14,36712806	8,525142737
14	0,856012953	8,372197637	5,29283524	7,994835352	8,547856364
15	4,286007549	8,274037881	5,269635279	37,24295638	8,568038129
16	1,078066482	8,468327954	5,448691762	37,75798411	8,755815714
17	1,01894143	8,49477063	5,475987898	19,3007921	8,738334017
18	6,508743408	8,734697848	5,651700012	43,01879516	8,944250033
19	1,188433588	8,764459295	5,731063872	43,11373415	9,042111406
20	1,249208519	8,872527101	5,919231256	52,10370396	9,20693622
Ortalama Süre	**1,699165114**	**8,12373906**	**5,035567654**	**15,26853064**	**8,232422499**

Her problem için gerçekleştirilen 30 çözüm içinde minimum değere sahip olan sonuç değerlerinin üretilmesinde harcanan süreler tüm algoritmalar için Çizelge 5.5'te listelenmiştir.

Çözümlerin süreleri göz önünde bulundurulduğunda ortalamaya en yakın çözüm sürelerinde olduğu gibi BT, TA ve KKO algoritmaları en yavaş sonuca ulaşan algoritmalardır. GA algoritmasının süreleri ise çok iyidir. Süre değerleri incelendiğinde GA ve PSO algoritmaları öne çıkarken minimum çözüm değerleri bu karşılaştırmaya eklendiğinde PSO algoritması ön plana çıkmaktadır.

Yukarındaki verilere dayanarak PSO ve BT algoritmalarının benzer koşullar altında diğer algoritmalara nazaran daha iyi sonuçlar ürettiği söylenebilir. Değişken ve kısıt sayısının az olduğu ilk problemlerde BT algoritması, değişken ve kısıt sayısının fazla olduğu son problemlerde ise PSO algoritması belirtilen koşullar altında daha iyi sonuçlara ulaşmışlardır. Çözümlerin süreleri göz önünde bulundurulduğunda iyi sonuç alınan algoritmalar arasında PSO algoritması BT algoritmasına nazaran daha kısa sürelerde sonuca ulaşmıştır. Diğer algoritmalara nazaran; PSO algoritması 13 problemde daha iyi sonuca ulaşırken BT algoritması 5 problemde daha iyi sonuca ulaşabilmiştir.

6. TARTIŞMA VE SONUÇ

Son elli yıldır, bilgisayarların gelişimine paralel olarak çok çeşitli optimizasyon algoritmaları geliştirilmiştir. Literatürde bu algoritmaların başarısının test edilmesine yönelik çok sayıda çalışma vardır. Bu çalışmalarda çeşitli test problemleri kullanılmaktadır. Ancak optimizasyon algoritmalarının performansını ölçmek için literatürde kullanılan test problemleri doğrusal olmayan ve çok fazla kısıdı olmayan problemlerdir. Oysa kısıt sayısının artması problem çözümünü daha da zorlaştırır. Bu tezde kısıt sayısının birden fazla olması durumunda optimizasyon algoritmalarının başarısı test edilmiştir. Bu amaçla literatürdeki test problemleri yerine, kesin çözümü simpleks yöntem ile kolayca hesaplanabilen doğrusal programlama problemleri tercih edilmiştir.

Doğrusal programlama problemleri, GA (Genetik Algoritma), KKO (Karınca Koloni Optimizasyonu), PSO (Parçacık Sürü Optimizasyonu), BT (Benzetilmiş Tavlama), TA (Tabu Arama) olmak üzere beş optimizasyon algoritmasının karşılaştırılması için kullanılmıştır. Problem çözümlerinde alınan sonuçlar, algoritmaların ulaşabildiği en iyi sonuç değil verilen başlangıç parametreleri ile ulaştıkları çözümlerdir. Uygun başlangıç değerleri, katsayılar, iterasyon ve sürü sayıları kullanıldığında daha iyi sonuçlara ulaşabilecekleri bilinmektedir.

Bu tez için tarafımızca yazılan KKO, PSO, TA algoritmaları ve matlab programı içersinde bulunan SA, GA algoritmaları olmak üzere beş optimizasyon algoritması kullanılmıştır. Doğruluklarının onayı için bu algoritmalar, Branin ve Sphere Fonksiyonları ile test edilmiştir. Daha sonra da iki değişkenli üç kısıtlı basit problemlerden 20 değişkenli dört kısıtlı karmaşık problemlere kadar toplam 20 rastgele üretilmiş problemle test edilmiştir. Her bir problem her algoritma için 30 kez çözdürülmüş ve sonuçlar kaydedilmiştir. Toplamda elde edilen 3000 adet veri kaydedilmiş ve veri analizi için oluşturulan programda sonuçlar değerlendirilmiştir. Veri analizi için kullanılan program, verileri istenilen ölçütlere göre süzerek

araştırılacak veri miktarını azaltmakta, istenilen özellikteki verileri işaretleyerek incelemeyi kolaylaştırmaktadır. Program ile bağıl hata oranları, ortalamaya en yakın değerler, minimum değerler ve bu değerler için harcanan süreler kolaylıkla çizelge haline getirilebilmektedir.

Yukarıda özelliklerinden bahsedilen 20 rastgele üretilmiş problemle test edilen algoritmalar basit ve karmaşık koşullarda sınanmışlardır. Bu sınamalar sonucunda elde edilen verilerden aşağıdaki çıkartımlar yapılabilir:

Doğrusal programlama problemlerinin çözümünde, PSO (Parçacık Sürü Optimizasyonu) ve BT (Benzetilmiş Tavlama) algoritmaları; GA (Genetik Algoritma), KKO (Karınca Koloni Optimizasyonu) ve TA (Tabu Arama) algoritmalarına nazaran daha iyi sonuçlara ulaşmayı başarmışlardır. Gerçekleştirilen çözümler sonucunda Çizelge 5.2'de verilen ortalamaya en yakın değerlen incelendiğinde KKO diğer algoritmalara göre daha iyi bir sonuca ulaşamamışken GA ve TA 1, BT 8, PSO 10 problemin çözümünde daha iyi sonuca ulaşmıştır. Çizelge 5.4'te verilen minimum değerler incelendiğinde ise KKO ve TA diğer algoritmalara göre daha iyi bir sonuca ulaşamazken GA 2, BT 5, PSO 13 problemin çözümünde daha iyi sonuca ulaşabilmiştir.

KKO ve TA algoritmaları kesikli optimizasyon problemleri için geliştirildiğinden sürekli optimizasyon problemlerinde diğer algoritmalar kadar iyi sonuç gösterememektedirler. PSO algoritması ise sürekli optimizasyon problemleri için geliştirildiğinden daha iyi sonuçlara ulaşabilmiştir.

Sonuç olarak genelde çok kısıtlı sürekli optimizasyon problemlerinde özelde ise doğrusal programlama problemlerinin çözümünde, PSO algoritmasının daha iyi sonuçlar verdiği görülmüştür. Uygun koşullar altında ve makul sürelerde iyi çözümlere ulaşıldığı için; karmaşık ve çok kısıtlı doğrusal programlama problemlerinde bir metot olarak kullanılabileceği düşünülmektedir.

Bundan sonra yapılacak karşılaştırma çalışmalarında diğer optimizasyon algoritmaları da sınamaya katılarak daha kapsamlı bir sonuç çıkarımı yapılabileceği gözlemlenmiştir.

KAYNAKLAR

AFACAN, C., 2007, Kalite Yönetim Sistemi ve Stratejik Planlamada Kalite, Yüksek Lisans. Marmara Üniversitesi.

AKAY, B., 2009, Nümerik Optimizasyon Problemlerinde Yapayarı Kolonisi (Artificial Bee Colony) Algoritmasının Performans Analizi, Doktora, Erciyes Üniversitesi.

ALATAŞ, B., 2007, Kaotik Haritali Parçacik Sürü Optimizasyonu Algoritmalari Gelistirme, Doktora, Fırat Üniversitesi.

ALTAY, E., 1990, Doğrusal Programlama Tekniği ile Ürün Karması Probleminin Çözümü ve T.C. Milli Eğitim Bakanlığı Ders Aletleri Yapım Merkezinde Bir Uygulama, Yüksek Lisans, Gazi Üniversitesi.

AYDIN, A., 2009, Metasezgisel Yöntemlerle Uçak Çizelgeleme Problemi Optimizasyonu, Doktora, Marmara Üniversitesi.

BACK, T., HAMMEL, U., SCHEFEL, A., 1997, Evolutionary Computation Comments on The History and Current State. IEEE Trans. Evol. Comput., 1, 3-17.

BALI, Ö., 2009, Stokastik U-Tipi Montaj Hattı Dengeleme Problemleri İçin Parçacık Sürü Optimizasyonu, Doktora, Gazi Üniversitesi.

BAŞKAN, Ö., HALDENBILEN, S., CEYLAN, H., CEYLAN, Ha., 2009, A New Solution Algorithm For Improving Performance of Ant Colony Optimization, Applied Mathematics and Computation, 211, 75-84.

BAZARAA, M., JARVIS, J. J., 1977, Linear Programming And Network Flows, Canada, America: John Wiley – Sons, İnc, 0-471-06015-1.

BOLAT, B., EROL, O., İMRAK, C., 2004, Mühendislik Uygulamalarında Genetik Algoritmalar ve Operatörlerin İşlevleri, Mühendislik ve Fen Bilimleri Dergisi, 4, 264-271.

BOLAT, B., 2006, Asansör Kontrol Sistemlerinin Genetik Algoritma ile Simülasyonu, Doktora Tezi, Yıldız Teknik Üniversitesi.

CLERC, M., KENNEDY, J., 2002, The Particle Swarm—Explosion, Stability, and Convergence in a Multidimensional Complex Space, IEEE Transactions On Evolutionary Computation, 6 (1), 58-73.

COŞKUN, A., 2006, Genetik Algoritma Kullanılarak Kimyasal Maddelerin Deriden Geçiş Katsayılarının ve Molekül Yapılarının Bulunması, Doktora, Gazi Üniversitesi.

CURA, T., 2008a, Doğrusal Olmayan Küresel Optimizasyon Problemleri İçin Tabu Arama Algoritmasının Kullanılması, İstanbul Üniversitesi İşletme Fakültesi Dergisi , 37 (1) , 22-38.

CURA, T., 2008b, Modern Sezgisel Teknikler ve Uygulamaları, Papatya Yayıncılık, İstanbul, 978-975-6797-79-2.

DANTZIG, G., 1963, Linear Programming and Extensions, Princeton University Press, Princeton, 0691059136.

DORIGO, M., MANIEZZO, V., COLORNI, A., 1991, Positive Feedback As a Search Strategy, Technical Report 91-016, 1-20.

DORIGO, M., MANIEZZO, V., COLORNI, A., 1996, The Ant System:Optimization by a Colony of Cooperating Agents, IEEE Transactions on Systems, Man, and Cybernetics–Part B, 26 (1), 1-13.

DORIGO, M., CARO, G. D., GAMBARDELLA, L., 1999, Ant Algorithms for Discrete Optimization, Artificial Life, 5(2), 137-172.

DORIGO, M., BLUMB, C., 2005, Ant colony optimization theory: A survey, Theoretical Computer Science, 344 , 243 – 278.

EBERHART, R., SHI, Y., 2001, Partical Swarm Optimization: Developments, Applications and Resources, Proc. IEEE Int'l Conf. on Evolutionary Computation, 2001 Seoul Korea, Piscataway, NJ: IEEE Service Center, 81-86.

ELKAMCHOUCHI, H., WAGIB, M., 2001, Failure Restoration and Array Synthesis Using Genetic Algorithms, Eighteenth National Kadio Science Conference, 2001 Egypt: Mansoura Univ, IEEE Piscataway NJ, 123-130.

ERDOĞAN, Ş. Z., 2008, Kendini Klonlayan Karınca Kolonisi Yaklaşımıyla Optimal Yolun Bulunması, Doktora, Trakya Üniversitesi.

GAMBARDELLA L. M., DORIGO, M., 1995, Ant-Q: A Reinforcement Learning Approach To The Traveling Salesman Problem, Proceeding of ML-95, Twelfth Intern. Conf. on Machine Learning, 1995 Palo Alto, CA: Morgan Kaufmann, 252-260.

GAMBARDELLA, L. M., DORIGO, M., 1996, Solving Symmetric and Asymmetric TSPs by Ant Colonies, IEEE Conference on Evolutionary Computation (ICEC'96), 1996 Nagoya Japan, IEEE Press,622-627.

GLOVER, F., 1986, Future Paths For Integer Programming and Links to Artificial Intelligence, Comput. & Ops. Res., 13 (5), 533-549.

GOSS, S., ARON, S., DENEUBOURG, J. L., PASTEELS, J. M., 1989, Self-organized Shortcuts in the Argentine Ant, Naturwissenschaften, 76, 579- 581.

GUNTSCH, M., MIDDENDORF, M., 2002, A population based approach for ACO, In S. Cagnoni and et al., editors, Applications of Evolutionary Computing - EvoWorkshops 2002: EvoCOP, EvoIASP, EvoSTIM/EvoPLAN, 2279, 72-81.

GÜLCÜ, A., 2006, Yapay Zeka Tekniklerinden Genetik Algoritma ve Tabu Arama Yöntemlerinin Eğitim Kurumlarının Haftalık Ders Programlarının Hazırlanmasında Kullanımı, Yüksek Lisans, Marmara Üniversitesi.

KAYA, T., 2006, Genetik Algoritma ile Sayısal Filtre Tasarımı, Yüksek Lisans, Fırat Üniversitesi.

KENNEDY, J., EBERHART, R., 1995, Particle Swarm Optimization, Proc. IEEE Intl. Conf. on Neural Networks, 1995 Perth Australia, Piscataway NJ: IEEE Service Center, 1942-1948.

KENNEDY, J., 1997, The Particle Swarm: Social Adaptation of Knowledge, Proceedings of the IEEE International Conference on Evolutionary Computation, 1997 Indianapolis USA, Piscataway NJ: IEEE Press, 303-308.

KILIÇ, S., 2008, Bulanık Karar Ortamında Karınca Kolonisi Optimizasyonu Yöntemiyle Araç Rotalama, Doktora, İstanbul Teknik Üniversitesi.

KIRKPATRICK, S., GELATT, C. D., VECCHI, J. M., 1983, Optimization by Simulated Annealing, Science, 220 (4598), 671-680.

KÜÇÜKTEZCAN, F., 2008, Genetik Algoritma ile Optimize Edilmiş Bulanık Güç Sistemi Kararlı Kılıcısının Sistem Kararlılığına Etkisi, Yüksek Lisans, İstanbul Teknik Üniversitesi.

LU, W., 2003, Optimum Design of Cold-Formed Stell Purlins Using Genetic Algorithms, PhD Thesis ,Helsinki University of Techonology Laboratary of Steel Structunes Publications 25.

MAN, K., TANG, K., KWONG, S., 1996, Genetic Algorithms : Concepts and Applications, IEEE Trans on Endustrial Electronics , 43 (5), 519-534.

MASTORAKIS, N., GONOS, I., SWAMY, M., 2003, Design of Two-Dimensional Recursive Filters Using Genetic Algorithms, IEEE Transactions on Curcuits and Systems-I : Fundamental Theory and Applications , 50 (5), 634-639.

METROPOLIS, N., ROSENBLUTH, A. W., ROSENBLUTH, M. N., TELLER, A. H., TELLER, E., 1953, Equation of State Calculations by Fast Computing Machines, J. Chem. Phys., 21 (6), 1087–1092.

MITCHELL, M., 1999, An İntroduction To Genetic Algorithms, Cambiridge: MIT Pres, London England, 0−262−13316−4 (HB).

ÖZÇILOĞLU, M. M., 2009, Algılayıcı Ağlarda Gözlemnememe Olgusunun Doğrusal Programlama ile İncelenmesi, Yüksek Lisans, TOBB Ekonomi ve Teknoloji Üniversitesi.

ÖZTÜRK, A., 2007, Güç Sistemlerindeki Gerilim Kararlılığının Genetik Algoritma ile İncelenmesi, Doktora, Sakarya Üniversitesi.

PARK, J., PARK, Y., WON, J., LEE, K., 2000, An Improved Genetic Algorithm for Generation Expansion Planning, IEEE Transsaction on Power Systems, 15 (3), 916-922.

SRINIVAS, N., DEB, K., 1994, Multiobjective Optimization Using Nondominated Sbrting in Genetic Algorithms, Evol. Comput., 2 (3), 221-248.

STÜTZLE, T., HOOS, H., 1997, MAX–MIN Ant System and Local Search For The Traveling Salesman Problem, İn Proc. IEEE Int. Conf. Evol. Comput., 1997 Indianapolis Indiana USA, Piscataway NJ: IEEE Press, 309–314.

TAHA, H. A., 2007, Yöneylem Araştırması (Ş. A. Baray, & Ş. Esnaf, Çev.), Literatür Yayıncılık, İstanbul, 975-8431-28-5.

TERZI, S., SERIN, S., 2010, Asfalt Üstyapı Bakım Çalışmalarının Karınca Kolonisi Optimizasyonu İle Planlanması, ASYU 2010, 2010 Kayseri.

TOKSARI, M. D., 2006, Ant Colony Optimization for Finding the Global Minimum, Applied Mathematics and Computation, 176, 308-316.

TOZAN, A., 2007, Sensör Yerleştirme Probleminin Genetik Algoritma ve Parçacık Sürü Optimizasyonu ile Çözümü, Yüksek Lisans, Gebze Yüksek Teknoloji Enstitüsü.

WOOK, C., RAMAKRISHNA, R.S., 2003, Elitism-Based Compact Genetic Algorithms, IEEE, Transactions on Evolutionary Computation, 7 (4) , 367-385.

ÖZGEÇMİŞ

Kişisel Bilgiler

Soyadı, adı : KÜÇÜKKÜLAHLI, Enver

Uyruğu : T.C.

Doğum tarihi ve yeri : 29.07.1983 Tosya

Medeni hali : Bekar

Telefon : 0 (380) 542 00 99

Faks : 0 (380) 542 00 98

e-mail : enverkucukkulahli@duzce.edu.tr, enver@enverk.com

Eğitim

Derece	Eğitim Birimi	Mezuniyet tarihi
Lisans	Ondokuz Mayıs Üniversitesi Bilgisayar ve Öğretim Teknolojileri Eğitimi	2006
Lise	Tosya Anadolu Lisesi	2001

İş Deneyimi

Yıl	Yer	Görev
2007-Halen	Düzce Üniversitesi	Öğretim Görevlisi
2006-2007	Halıcı Yazılım Yapı Kredi Bankası A.Ş	Eğitim Teknoloğu

Yabancı Dil

İngilizce

Hobiler

Bilgisayar teknolojileri, Seyahat etmek, Fotoğrafçılık, Resim yapmak, Dalış, Model uçak- araba yapımı