T.C.
PAMUKKALE ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ
BİLGİSAYAR MÜHENDİSLİĞİ ANABİLİM DALI

NUMERİK VE PARÇACIK SÜRÜ OPTİMİZASYONU YÖNTEMLERİ BİRLEŞTİRİLEREK KURGULANMIŞ YENİ BİR OPTİMİZASYON ALGORİTMASI

YÜKSEK LİSANS TEZİ

MEHMET CENGİZ

DENİZLİ, AĞUSTOS - 2013

T.C.
PAMUKKALE ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ
BİLGİSAYAR MÜHENDİSLİĞİ ANABİLİM DALI

NUMERİK VE PARÇACIK SÜRÜ OPTİMİZASYONU YÖNTEMLERİ BİRLEŞTİRİLEREK KURGULANMIŞ YENİ BİR OPTİMİZASYON ALGORİTMASI

YÜKSEK LİSANS TEZİ

MEHMET CENGİZ

DENİZLİ, AĞUSTOS - 2013

YÜKSEK LİSANS TEZ ONAY FORMU

Pamukkale Üniversitesi Fen Bilimleri Enstitüsü 111281002 nolu öğrencisi Mehmet CENGİZ tarafından hazırlanan "**NUMERİK VE PARÇACIK SÜRÜ OPTİMİZASYONU YÖNTEMLERİ BİRLEŞTİRİLEREK KURGULANMIŞ YENİ BİR OPTİMİZASYON ALGORİTMASI**" başlıklı tez tarafımızdan okunmuş, kapsamı ve niteliği açısından bir Yüksek Lisans tezi olarak kabul edilmiştir.

Tez Danışmanı : (Jüri Başkanı) **Yrd. Doç. Dr. Emre ÇOMAK (PAÜ)**

Jüri Üyesi : **Prof. Dr. Serdar İPLİKÇİ (PAÜ)**

Jüri Üyesi : **Doç. Dr. Kadir KAVAKLIOĞLU (PAÜ)**

Pamukkale Üniversitesi Fen Bilimleri Enstitüsü Yönetim Kurulu'nun 28.08.2013. tarih ve ...28/13.... sayılı kararıyla onaylanmıştır.

Fen Bilimleri Enstitüsü Müdürü
Prof. Dr. Nuri KOLSUZ

Bu tezin tasarımı, hazırlanması, yürütülmesi, araştırmalarının yapılması ve bulgularının analizlerinde bilimsel etiğe ve akademik kurallara özenle riayet edildiğini; bu çalışmanın doğrudan birincil ürünü olmayan bulguların, verilerin ve materyallerin bilimsel etiğe uygun olarak kaynak gösterildiğini ve alıntı yapılan çalışmalara atfedildiğine beyan ederim.

Mehmet CENGİZ

ÖZET

NUMERİK VE PARÇACIK SÜRÜ OPTİMİZASYONU YÖNTEMLERİ BİRLEŞTİRİLEREK KURGULANMIŞ YENİ BİR OPTİMİZASYON ALGORİTMASI
YÜKSEK LİSANS TEZİ
MEHMET CENGİZ
PAMUKKALE ÜNİVERSİTESİ FEN BİLİMLERİ ENSTİTÜSÜ
BİLGİSAYAR MÜHENDİSLİĞİ ANABİLİM DALI
(TEZ DANIŞMANI: YRD. DOÇ. DR. EMRE ÇOMAK)
DENİZLİ, AĞUSTOS - 2013

Bu çalışma doğrusal olmayan problem çözümleri için Broydon-Fletcher-Goldfarb-Shanno (BFGS) ve Parçacık Sürü Optimizasyonu (PSO) yöntemleri temelli melez bir optimizasyon algoritması önermektedir. Literatürde kullanılan optimizasyon test fonksiyonlarını çözmek için BFGS içine PSO yerleştirilmiştir. Kulanılan foknsiyonlar, Rosenbrock vadisi, Rastrigin, Griewangk fonksiyonlarıdır. Bunu gerçekleştirmek için başlangıç noktası öncelikle BFGS ile aranmakta ve en iyi çözüm, daha iyi sonuçlar için, PSO'ya gönderilmektedir. Bu işlem karşılıklı paslaşmalarla sonlandırma koşulları sağlanana kadar devam etmektedir.

Ardından karşılaştırma yapmak için ikinci bir algoritma geliştirilmiştir. İkinci algoritmada ise başlangıç çözümü PSO ile bulunmuş ve en iyi çözüm BFGS'ye gönderilmiştir. En sonunda da algoritmamız ikinci geliştirilen algoritma ve standart PSO algoritması ile karşılaştırılmıştır.

Sayısal deneyler bütün sonuçların optimum noktalara çok yaklaştığını göstermiştir. Hız ve alınan sonuçlar karşılaştırıldığında geliştirdiğimiz algoritma ile karşılaştırma algoritmasının hemen hemen aynı olduğu tespit edilmiştir. Öte yandan standart PSO daha hızlı olmakla birlikte algoritmamız çok daha iyi sonuçlar bulmuştur. Sonuç olarak önerilen algoritma doğrusal olmayan problemlerle ilgilenirken kullanılabilecek etkili bir melez algoritma olmuştur.

ANAHTAR KELİMELER: BFGS, PSO, Optimizasyon, Melez Algoritmalar, Sezgisel Yöntemler, Numerik Yöntemler

ABSTRACT

A NEW HYBRID ALGORITHM BUILT BY COMBINING NUMERIC AND PARTICLE SWARM OPTIMIZATION METHODS
MSC THESIS
MEHMET CENGİZ
PAMUKKALE UNIVERSITY INSTITUTE OF SCIENCE
COMPUTER ENGİNEERİNG
(SUPERVISOR: ASSIST. PROF. DR. EMRE ÇOMAK))
DENİZLİ, AUGUST 2013

This work proposes a hybrid optimization algorithm based on BFGS method and PSO to solve nonlinear programs. The algorithm integrates the BFGS into PSO to solve test functions for optimization. These functions are Rosenbrock's Valley, Rastrigin's function, Griewangk's function. In doing so, on initial search is firstly attempted by BFGS and then the best solution is passed on to PSO for further investigation. This sequence as a whole is iterated as many times as required to meet the stopping conditions.

After that we developed a second algorithm to compare. Towards this end, on initial search is firstly attempted by PSO and then the best solution is passed on to BFGS. Finally, we compare our algorithm with standard PSO and our second algorithm.

The numerical experiments show that all experimients' results are close to the optimum point. The results and spped of first and second algorithm are almost same. On the other hand, standart PSO is faster than our first algorithm although first algorithm's results are better than standart PSO's. As a result proposed algorithm makes an effective use of hybrid framework when dealing with nonlinear equality contraints.

KEYWORDS: BFGS, PSO, Optimization, Hybrid Algorithms, Heuristic Methods, Numeric Methods

İÇİNDEKİLER

Sayfa

ÖZET ..... **i**
ABSTRACT ..... **ii**
İÇİNDEKİLER ..... **iii**
ŞEKİL LİSTESİ ..... **v**
TABLO LİSTESİ ..... **vi**
ÖNSÖZ ..... **vii**
1. GİRİŞ **1**
2. LİTERATÜR BİLGİSİ **5**
2.1 Doğrusal Olmayan Numerik Programlama Çalışmaları 5
2.2 Sezgisel Yöntem Çalışmaları 6
2.3 Test Fonksiyonları 7
3. DOĞRUSAL OLMAYAN NUMERİK PROGRAMLAMA **8**
3.1 Bir Boyutlu Numerik Optimizasyon 8
3.1.1 Türev Bilgisi Kullanan Yöntemler 9
3.1.2 Türev Bilgisi Kullanmayan Yöntemler 9
3.1.2.1 Altın Bölme Yöntemi (ABY) 9
3.2 Çok Boyutlu Numerik Optimizasyon 11
3.2.1 Gradyant Olmayan Yöntemler 11
3.2.2 Grandyant Yöntemler 11
3.2.2.1 Broydon-Fletcher-Goldfarb-Shanno (BFGS) Yöntemi 11
3.2.2.2 Davidon-Fletcher-Powell (DFP) Yöntemi 12
4. SEZGİSEL PROGRAMLAMA **14**
4.1 Evrimsel Hesaplama Algoritmaları 14
4.1.1 Genetik Algoritmalar 15
4.1.2 Diferansiyel Gelişim Algoritması 16
4.2 Gelişim Algoritmaları 17
4.2.1 Tabu Araştırma Algoritmaları 17
4.2.2 Isıl İşlem Algoritması 18
4.3 Sürü Temelli Algoritmalar 19
4.3.1 Karınca Kolonisi Algoritması 19
4.3.2 Yapay Arı Kolonisi Algoritması 20
4.3.3 Parçacık Sürü Optimizasyonu (PSO) Algoritması 21
4.3.3.1 Parçacık Sürü Optimizasyonu Kavramları 21
5. GELİŞTİRİLEN YÖNTEM **25**
5.1 Açıklamalar 25
5.2 Akış Şeması 26
5.2.1 Altın Bölme Yöntemi Akış Diyagramı 26
5.2.2 Broydon-Fletcher-Goldfarb-Shanno Yöntemi Akış Diyagramı 27
5.2.3 Parçacık Sürü Optimizasyonu Akış Diyagramı 28
5.2.4 Birleştirilmiş Akış Diyagramı 28
5.3 Algoritma 29
5.4 Karşılaştırma İçin Oluşturulan Algoritma 30
6. UYGULAMALAR **32**
6.1 Kullanılan Test Fonksiyonları 32
6.2 Sonuçlar 36

7. TARTIŞMALAR VE SONUÇ 43
8. KAYNAKLAR 45
9. ÖZGEÇMİŞ 47

ŞEKİL LİSTESİ

Sayfa

Şekil 1.1: VE Mantık Kapısı ve Doğrusal Olarak Ayrıştırılması 1
Şekil 1.2: ÖZEL VEYA Mantık Kapısı ve Doğrusal Olarak Ayrıştırılması 2
Şekil 4.1: Parçacık Hareketi 23
Şekil 5.1: ABY Akış Diyagramı 26
Şekil 5.2: BFGS Akış Diyagramı 27
Şekil 5.3: PSO Akış Diyagramı 28
Şekil 5.4: Birleştirilmiş Akış Diyagramı 29
Şekil 6.1: De Jong Fonksiyonu 32
Şekil 6.2: Rastrigin Fonksiyonu 33
Şekil 6.3: Griewangk Fonksiyonu 34
Şekil 6.4: Rosenbrock Fonksiyonu 34
Şekil 6.5: Schwefel Fonksiyonu 35
Şekil 6.6: Ackley Fonksiyonu 36

TABLO LİSTESİ

Sayfa

Tablo 6.1: Rastrigin İçin Üretilen Rastgele Başlangıç Değerleri.....37
Tablo 6.2: Rastrigin Fonksiyonu Çözümleri ve Çözüm Süreleri.....37
Tablo 6.3: Rosenbrock İçin Üretilen Rastgele Başlangıç Değerleri 39
Tablo 6.4: Rosenbrock Fonksiyonu Çözümleri ve Çözüm Süreleri 39
Tablo 6.5: Griewangk Fonksiyonu Çözümleri ve Çözüm Süreleri.....41
Tablo 6.6: Griewangk Fonksiyonu Çözümleri ve Çözüm Süreleri.....41

ÖNSÖZ

Yüksek lisans eğitimimin başlangıcından tez çalışmasının sonuçlandırılmasına kadar olan her aşamasında beni yönlendiren, değerli bilgilerini ve zamanını esirgemeyen tez danışmanım Sayın Yrd. Doç. Dr. Emre ÇOMAK'A sonsuz teşekkürlerimi sunarım.

Kendisinden aldığım Bilgisayar Bilimlerinde İleri Teknikler dersi ile optimizasyon konusuna yönelmemi sağlayan ve tez çalışmam süresince desteklerini esirgemeyen, tez jüri üyesi Sayın Doç. Dr. Kadir KAVAKLIOĞLU'NA teşekkür ederim. Tez çalışması süresince ders notlarını referans aldığım ve algoritmalarını kullandığım, tez jüri üyesi Sayın Prof. Dr. Serdar İPLİKÇİ'YE teşekkürü borç bilirim.

Programlama ortamıyla ilgili karşılaştığım sorunları çözerken yardımlarını esirgemeyen Sayın Araş. Gör. Turgay BATBAT ve tez çalışmasana başlarken fikir alışverişinde bulunduğum Sayın Araş. Gör. Ahmet Şakir DOKUZ'A teşekkür ederim.

Çalışmalarım sırasında manevi desteğini benden esirgemeyen aileme ve tez yazım sürecinde teknik imkanlar sağlayan Sayın Cansu TOPKAYA'YA sonsuz teşekkürlerimi borç bilirim.

1. GİRİŞ

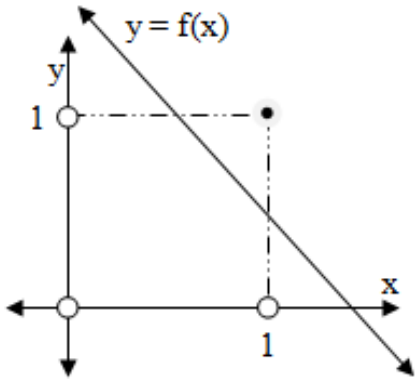
Belirli sınırlamaları sağlayacak şekilde, bilinmeyen parametre değerlerinin bulunmasını içeren herhangi bir problem, optimizasyon problemi olarak adlandırılabilir [1]. Karşılaşılan problemleri çözmek için birçok yol vardır ama hepsi etkin bir sonuç değildir. Hatta kimi çözümler daha maliyetli olmaktadır. İşte bu sebeple optimizasyona ihtiyaç duyulmaktadır.

Optimizasyon uygulanabilen problemler genellikle ikiye ayrılır ve her iki tip için de farklı çözüm yöntemleri kullanılır. Bunlar doğrusal ve doğrusal olmayan problemlerdir.

Doğrusal problemler basit yapılı ve tek katlı tasarım değişkenlerine sahiptir. Koordinat düzleminde veya bir grafikte çözüm noktaları tek bir doğru ile ayrıştırılabilirler. Denklem 1.1'de doğrusal bir denklem örneği bulunmaktadır.

$$f(x,y) = x + 2y + 5xy = 0 \qquad (1.1)$$

Şekil 1.1'de ise koordinat düzleminde VE mantık kapısı verilmiş olup doğrusallığı gösterilmiştir. Tek bir doğru ile 0 noktaları ve 1 noktası ayrıştırılabilmektedir. Gösterimde 0 noktaları için içi boş daireler, 1 noktası için de içi siyah daire kullanılmıştır.

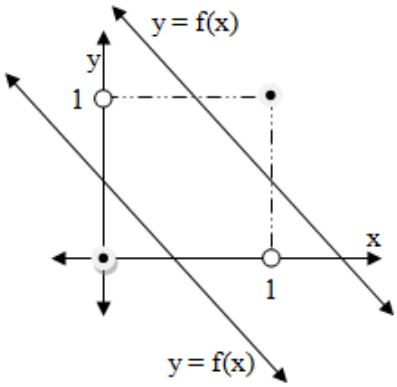


Şekil 1.1: VE Mantık Kapısı ve Doğrusal Olarak Ayrıştırılması

Doğrusal olmayan problemler tasarım değişkeni çok katlı ve/veya trigonometrik, diferansiyel vb gibi doğrusal olmayan başka ifadeler içeren problemlerdir. Doğrusal denklemlerin aksine koordinat düzleminde veya grafiksel gösterimde çözüm noktaları tek bir doğruyla ayrıştırılamazlar. Denklem 1.2'de doğrusal olmayan bir denklem örneği verilmiştir.

$$f(x, y) = x2 + 2y3 + 5xy = 0 \qquad (1.2)$$

Şekil 1.2'de ise koordinat düzleminde ÖZEL VEYA mantık kapısı verilmiştir. Şekilden de görüldüğü gibi ancak ya en az iki doğru ya da bir eğri ile 0 noktaları ve 1 noktaları ayrıştırılabilir. Gösterimde 0 noktaları için içi boş daireler, 1 noktaları içinde içi siyah daireler kullanılmıştır.



Şekil 1.2: ÖZEL VEYA Mantık Kapısı ve Doğrusal Olarak Ayrıştırılması

Bu çalışmada doğrusal olmayan problem çözümleri üzerinde durulacak olup doğrusal problemlere daha fazla değinilmeyecektir.

Doğrusal olmayan problem çözümleri için kullanılan yöntemler kabaca iki başlık altında toplanabilir: Doğrusal olmayan programlama ve sezgisel yöntemler.

Doğrusal olmayan programlama yöntemleri, matematiksel ifadeler ve/veya türevlere dayalı çözümler içeren yöntemlerdir. Çoğu bu tip yöntem küresel çözümü garanti edemezken yerelde kesin çözümler sağlayabilir. Bu sayede, bazı mühendislik uygulamalarındaki, hata payı en küçüğe indirgenmiş olur. Zayıf kaldığı noktalardan bir tanesi ise esnek olmayışıdır. Farklı tip problemler için tekrar düzenleme

gerektirmektedir. Doğrusal olmayan programlama, çözüm yöntemlerine göre ikiye ayrılır: Analitik ve numerik yöntemler.

Analitik çözüm yöntemleri alt alta matematiksel denklemlerin çözülmesine dayalı yöntemlerdir. Denklem sistemlerinin çözümü gerektiği için günümüzde kullanılan bilgisayar teknolojileri çözüm sürecinde yetersiz kalmaktadır. Kağıt ve kalem kullanılarak çözüm yapılır, sonuç ise bilgisayar teknolojileri sayesinde bulunabilir. Bu çalışmada analitik yöntemler kullanılmayacağı için daha fazla üzerinde durulmayacaktır.

Numerik çözüm yöntemleri türevlerle ve/veya adım adım yaklaşma kuralları ile çözülen tamamen sayısal değerlerin denenmesine yönelik çözüm yöntemleridir. Bu çalışmada, çok boyutlu lineer olmayan numerik optimizasyon yöntemlerinden Variable Metric Method (VMM) ailesi kullanılmıştır. VMM ailesi fonksiyonun geçmişte kullanılan bütün yönlerine ait bilgileri tuttuğu "Metric" adı verilen n x n boyutlarında bir matris kullanır. Bu matris için başlangıç olarak simetrik ve pozitif tanımlı bir matris atanır. Çözüme ulaşıldığında metric matrisi genellikle "Hessian Matrisi" ile ilişkili olur.

VMM ailesi optimizasyon yöntemleri temelde şu adımlardan oluşur [2]:

i. Bir başlangıç noktası ile başla.
ii. Adım yönü seç.
iii. Yeni yöne göre bulunan noktanın konumunu belirle.
iv. Amaç fonksiyonunu ve gradyant değerini hesapla.
v. Metric matrisini güncelle.
vi. Yeni metric'i ve fonksiyonun minimumluğunu dene.

Sezgisel yöntemler ise doğal olguların modellenerek problem çözümüne uyarlanması ile geliştirilmiş yapay zeka teknikleridir. Çok esnek yapıda olup numerik yaklaşımlara göre daha hızlıdır. Küresel çözümlere yaklaşma konusunda çok yetenekli olmalarına rağmen en büyük olumsuzlukları kesin çözüm bulmayı garanti edememeleridir.

Bu çalışmada numerik yöntemlerden biri olan Broydon-Fletcher-Goldfarb-Shanno (BFGS) yöntemi, sezgisel yöntem olan, Parçacık Sürü Optimizasyonu (PSO)

ile birleştirilerek yeni bir melez optimizasyon tekniği oluşturulmuştur. Bu sayede numerik yöntemlerin kesinliğini sezgisel yöntemlerin hızıyla birleştirilmesi amaçlanmıştır.

Çalışmanın ikinci bölümünde bu alanda yapılmış çeşitli çalışmalardan bahsedilmiştir. Üçüncü bölümde konumuzla ilgili olarak doğrusal olmayan numerik programlama ve sınıflandırılması izah edilmiş olup özellikle kullanılacak yöntemler vurgulanmıştır. Dördüncü bölümde sezgisel yöntemler çalışmanın amaçları doğrultusunda sınıflandırılmış ve her bir sınıfa örnek yöntemler verilmiştir. Her örneğin de genel algoritmaları eklenmiştir. Beşinci bölümde çalışmamızdaki numerik ve sezgisel yöntemlerin kullanımı ve üretilen algoritma anlatılmıştır. Altıncı bölümde ise çalışmadan elde ettiğimiz sonuçlar ve kullanılan test fonksiyonları da eklenmiştir. Yedinci ve son bölümümüzde de elde ettiğimiz sonuçların değerlendirilmesine yer verilmiştir.

2. LİTERATÜR BİLGİSİ

Bilgisayar teknolojilerinin tarihi boyunca geliştiği noktalar incelendiğinde bu teknolojilere duyulan ihtiyaç her zaman daha fazla olmuştur. Bu durumdan dolayı da, sistemin daha verimli kullanılabilmesi için teknolojinin sürekli iyileştirilmesine gerek duyulmaktadır. Bu iyileştirmeyi yapabilmek için de, her geçen gün farklı optimizasyon algoritmaları üretilmektedir. Literatürde çözüm yollarına, yaklaşım anlayışlarına, amaçlarına vb. göre farklı sınıflandırılmış onlarca optimizasyon tekniği bulunmaktadır.

2.1 Doğrusal Olmayan Numerik Programlama Çalışmaları

Doğrusal olmayan numerik programlama tekniklerinden VMM ailesi kullanılmıştır. Bu yöntemler yoğun türev bilgisi gerektirmektedir. Bu durumda türevi alınabilen amaç fonksiyonuna ihtiyaç duyulmaktadır. Örneğin amaç fonksiyonu bulunmayan Gezgin Satıcı Problemlerinin (GSP) çözümünde VMM kullanılamamaktadır.

Literatürde VMM ailesi, bizim çalışmamızda olduğu gibi, başka tekniklerle birleştirilmiş, o tekniklere yardımcı olarak kullanılmıştır. Örneğin;

- Nawi, Ransing M., Ransing R.'nin yaptığı "An Improved Learning Algorithm Based on BFGS Method For Back Propagation Neural Nteworks" adlı çalışmada yapay sinir ağları ile kümeleme problemleri çözülmüştür. Burada yapay sinir ağının eğitilmesi için BFGS yöntemi kullanılmıştır [19].

- Xia, Chan ve Liu' nun yaptığı "Improved BFGS Method for Optimal Power Flow Calculation with Transient Stability Constraints" adlı çalışmada en iyi enerji akımı için BFGS yöntemi geliştirilip kullanılmıştır [20].

- Semeter ve Medillo'nun yaptığı "A Nonlinear Optimization Technique for Ground-Based Atmospheric Emission Tomography" adlı çalışmada

yeni bir lineer olmayan optimizasyon tekniği geliştirilmiş ve atmosferik salınım tomografisi çıkarılmıştır [21].

Bunun haricinde literatürde birçok geliştirilmiş VMM ailesine dahil veya dahil olmayan yeni optimizasyon teknikleri bulmak mümkündür.

2.2 Sezgisel Yöntem Çalışmaları

Doğal süreçte bütün olgular kaotik durumdan kurtulup durağan ve sakin konuma geçmeye çalışmaktadır. Bu kaotik durumlar gerek yiyecek bulma davranışında, gerek atomik boyutta hareketlerde ve gerekse de üreme davranışlarında karşımıza çıkmaktadır. Doğal süreçler bu tür kaotik davranışları olabilecek en uygun ve en az maliyetle çözmeye çalışırlar. Buradan yola çıkarak literatürde günlük hayatta karşılaşılan özellikle sanayi ağırlıklı olan problemleri çözmek için doğal olguları kullanan algoritmalar geliştirilmiştir. Bu problemlerin en çok bilinenlerinden biri mevcut noktalar arasında en kısa yolla ulaşım yapmayı amaçlayan GSP, diğeri ise en az maliyetle en çok üretim yapmayı amaçlayan üretim problemleridir. Öte yandan literatürde sezgisel yöntemler başka tekniklerle birlikte kullanılarak geliştirmeler de yapılmıştır. Örneğin;

- Bell ve McMullen'in yaptığı "Ant Colony Optimization Techniques for The Vehicle Routing Problem" adlı çalışmada karıncaların yem bulma davranışları modellenerek araç rotalama problemi çözülmüştür [7].

- Keskintürk'ün yaptığı "Diferansiyel Gelişim Algoritması" adlı çalışmasında, diferansiyel gelişim algoritması sezgiseli anlatılmış ve bu algoritma literatürde bulunan test fonksiyonlarının çözümünde kullanılmıştır [5].

- Szeto, Wu ve Ho'nun yaptığı "An Artifical Bee Colony Algorithm for The Capacitated Vehicle Routing Problem" adlı çalışmada araç rotalama problemi bu sefer de arıların polen kaynağı olan çiçeklerin yerini göstermek için yaptığı dansın modellenmesiyle oluşturulmuş yapay arı kolonisi sezgiseli kullanılarak çözülmüştür [10].

- Cura'nın yaptığı "Doğrusal Olmayan Küresel Optimizasyon Problemleri için Tabu Arama Algoritmasının Kullanılması" adlı çalışmasında kötü sonuçların elenerek bir yasak listesine konulması temeline dayalı tabu arama sezgiseli kullanılarak doğrusal olmayan optimizasyon problemleri çözülmüştür [6].

- Perez ve Behdinan'ın yaptığı "Particle Swarm Approach for Structural design Optimization" adlı çalışmada kuş ve balık sürülerinin yem bulmak veya tehlikeden kaçmak için kullandıkları yöntemlerin modellenmesi ile geliştirilen parçacık sürü optimizasyon teknikleri kullanılarak yapısal tasarım optimizasyonu çözülmüştür [15].

Literatürde bunların haricinde başka problemlerin çözümünde de sezgisel yöntemlerin kullanımı anlatılmıştır. Bazı çalışmalarda sezgiselleri geliştirmeye yönelik yeni yaklaşımlar da bulunmaktadır.

2.3 Test Fonksiyonları

Literatürdeki test fonksiyonları hepsi sürekli olmakla birlikte dört başlık halinde toplanmaktadır. Bunlar [4]:

i. Tek model, dışbükey, çok boyutlu.
ii. Çok modelli, az sayıda yerel ekstremum içeren iki boyutlu.
iii. Çok modelli, çok sayıda yerel ekstremum içeren iki boyutlu.
iv. Çok modelli, çok sayıda yerel ekstremum içeren çok boyutlu.

Bu test fonksiyonları geliştirilen optimizasyon tekniğinin sınanması için karmaşık matematiksel denklemler içeren ve belli kısıtlamalar altında çözülen fonksiyonlardır.

Çalışmamızda doğrusal olmayan problemler için önerilecek çözümler üzerinde durulmuş olup bir numerik ve bir de sezgisel yöntem melezlenmiştir. Aşağıda bu alanlarda yapılan ve bu yaklaşımları test etmek maksadıyla kullanılan literatürde mevcut yayınlar ve yaklaşımlara değinilmiştir.

3. DOĞRUSAL OLMAYAN NUMERİK PROGRAMLAMA

Doğrusal olmayan programlama, ilgilenilen problemde amaç veya kısıt fonksiyonlarından en az birisinin doğrusal olmama durumundaki optimizasyon problemlerinin çözüm sürecine denir. Doğrusal olmayan problemler mühendislik uygulanmalarında sıkça karşımıza çıkmaktadır. Kısıtlı sigmoidal/hiperbolik tanjant problemleri bu tür problemlerin özel örnekleridir.

Doğrusal olmayan numerik programlama teknikleri çalışılmak istenen alana göre farklı sınıflandırılabilir. Bu kısımda tasarım değişkeni sayısına bağlı olarak sınıflandırma yapılması tercih edilmiştir: Bir boyutlu numerik optimizasyon, çok boyutlu numerik optimizasyon.

3.1 Bir Boyutlu Numerik Optimizasyon

Bir boyutlu nümerik optimizasyon teknikleri tek tasarım değişkeni ile kurulan denklem sistemlerini çözmek üzerine geliştirilmiş yöntemlerdir. Matematiksel olarak standart formatı aşağıdaki gibidir [3].

$$\begin{aligned} &\text{minimize/maksimize } f(x) && \text{(en düşük/en büyük yapılacak amaç fonksiyonu)} \\ &x_{alt} \leq x \leq x_{üst} && \text{(tasarım değişkeni kısıtları)} \end{aligned} \tag{3.1}$$

Bir boyutlu numerik optimizasyon, çok boyutlu numerik optimizasyon probleminin çözümü sırasında adım aralığının belirlenmesinde kullanılır. Çok boyutlu nümerik optimizasyonda daha sonra da görüleceği gibi genel güncelleme kuralı,

$$x_{k+1} = x_k + \mathbf{sp} \tag{3.2}$$

şeklinde olup burada **p** arama yönü, **s** adım aralığıdır. Problemlerde güncelleme yapılırken önce uygun bir arama yönü belirlenir. Arama yönü belirledikten sonra, uygun bir adım aralığı seçimi artık bir boyutlu nümerik optimizasyon problemine dönüşür [3].

Bir boyutlu numerik optimizasyon tekniklerinde sınıflandırmalar kullanılacak alana göre yapılacağı gibi bu çalışmada türev bilgisi üzerine sınıflandırma yapılması tercih edilmiştir. Buna göre sınıflandırma, türev bilgisi kullanan ve türev bilgisi kullanmayan teknikler olarak ikiye ayrılır.

3.1.1 Türev Bilgisi Kullanan Yöntemler

Bütün nümerik tekniklerde olduğu gibi rastgele bir değerden çözüme başlanır. Türevin sıfır olduğu noktada en küçük veya en büyük noktaya ulaştığımız için bulunan her değer fonksiyonun türevi ile karşılaştırılır. Türev bilgisi kullanan yöntemler genel olarak aşağıdaki özelliklere sahiptir [3]:

i. Geometrik bir temele sahiptirler.
ii. Doğrusal olarak açılmış Taylor serilerini kullanırlar.
iii. İteratiftirler.
iv. Karesel olarak yakınsarlar.

3.1.2 Türev Bilgisi Kullanmayan Yöntemler

Bu tip yöntemlerde genel olarak fonksiyonun belli bir aralığında rastgele değerler üretilir. Bu değerler belli kurallara göre adım adım birbirlerine yaklaşırlar. Her yeni adımda bulunan değerler fonksiyonda yerine konularak sonuç değerlendirilir ve istenilen noktaya gelince durulur. Bu çalışmada türev bilgisi gerektirmeyenler arasında en cazip ve uygulaması kolay yöntemlerden biri olan Altın Bölme Yöntemi kullanılmıştır.

3.1.2.1 Altın Bölme Yöntemi (ABY)

Altın oran (AO), matematik ve sanatta, bir bütünün parçaları arasında gözlemlenen, uyum açısından en yetkin boyutları verdiği sanılan geometrik ve sayısal bir oran bağıntısıdır. Oranın ifadesi aşağıdaki gibidir:

$$AO=(1+\sqrt{5})/2 \quad (3.3)$$

Bu değer de yaklaşık olarak 1,61803'e denk gelmektedir.

Altın bölme yönteminde ise altın orandan elde edilmiş 0,38197 değeri kullanılmaktadır. Bu yöntem aralığı uçlardan aynı oranda daralttığı için fonksiyonun şekil ve süreklilik özelliklerinden bağımsız çalışır. En önemli özelliği de çözüme belli bir tolerans ile ulaşmak için gerekli iterasyon sayısını önceden tahmin edilebilmesidir. Algoritması aşağıdaki gibidir [3]:

Adım 1: x^{low} ve x^{up} belirle

$\tau = 0.38197$ (Altın Oran'dan)

ε = tolerans = $(\Delta x)_{final} / (x^{up} - x^{low})$

N = iterasyon sayısı = $-2.078 * \ln \varepsilon$

k = 1

Adım 2: $x_1 \leftarrow (1 - \tau)\, x^{low} + \tau\, x^{up}$; $f1=f(x_1)$

$x_2 \leftarrow \tau\, x^{low} + (1 - \tau)\, x^{up}$; $f2=f(x_2)$

Adım 3: Eğer k<N ise

Eğer f1>f2 ise

$x^{low} \leftarrow x1$; $x1 \leftarrow x2$; $f1 \leftarrow f2$

$x_2 \leftarrow \tau\, x^{low} + (1 - \tau)\, x^{up}$; $f2 = f(x_2)$

Eğer f2>f1 ise

$x^{up} \leftarrow x2$; $x2 \leftarrow x1$; $f2 \leftarrow f1$

$x_1 \leftarrow (1 - \tau)\, x^{low} + \tau\, x^{up}$; $f1 = f(x_1)$

$k \leftarrow k+1$

Adım 3'e git.

3.2 Çok Boyutlu Numerik Optimizasyon

3.2.1 Gradyant Olmayan Yöntemler

Gradyant olmayan yöntemler temelde sezgisel yöntem barındıran tekniklerdir. Bu tip yöntemler ileride bahsedileceği gibi türev bilgisi kullanmadan doğal yöntemler veya farklı modellemeler kullanarak yeni değerler üretmekte ve çözüme ulaşmaktadır. Bunlara örnek olarak genetik algoritmalar, yapay sinir ağları verilebilir.

3.2.2 Grandyant Yöntemler

Bu tip yöntemler adından da anlaşılacağı gibi kısmi türevler kullanarak çözüme ulaşan yöntemlerdir. Gradyant yöntemlerin çoğu Hessian matrisine yakınsar veya Hessian matrisine dönüşür. Bu yöntemlerde türev yönü ve adım aralığı önemlidir. Fonksiyonun yönü kullanılan algoritmaya göre farklı yöntemlerle bulunur. Adım aralığı ise, daha önce de belirtildiği gibi, bir boyutlu numerik optimizasyon yöntemlerini kullanılarak tespit edilecektir. Bu çalışmada kullanılan bir boyutlu numerik optimizasyon yöntemi ise altın bölme yöntemidir.

3.2.2.1 Broydon-Fletcher-Goldfarb-Shanno (BFGS) Yöntemi

Variable Metric Methods (VMM) ailesinin bir üyesi olup en çok tercih edilenidir. BFGS'nin diğer yöntemlerden farklı en büyük özelliği başlangıçta atanan metric matrisinin Hessian matrisinin kendisine yakınsaması ve hatta çözüme ulaşıldığında doğrudan Hessian matrisine eşit olmasıdır. Algoritması aşağıdaki gibidir [3]:

Adım 1: Başlangıç noktası = $\mathbf{x_1}$, metric = $\mathbf{A}$, iterasyon sayısı=N belirle.
Sonlandırma kriterleri ε_1, ε_2, ε_3 değerleri belirle.
k=1 ile iterasyonu başlat.

Adım 2: $\mathbf{x_k}$ noktasındaki gradyant vektör = $\nabla f(\mathbf{x_k})$ hesapla.

$\mathbf{A_k p_k} = -\nabla f(\mathbf{x_k})$

Bir boyutlu optimizasyon ile $f(\mathbf{x_k} + \mathbf{s p_k})$'yi minimum yapan $\mathbf{s_k}$'yi bul.

$\mathbf{x_{k+1}} = \mathbf{x_k} + \mathbf{s_k p_k}$

Adım 3: $\Delta f = f(\mathbf{x_{k+1}}) - f(\mathbf{x_k})$; $\Delta x = \mathbf{x_{k+1}} - \mathbf{x_k}$

Eğer $|\Delta f| \leq \varepsilon_1$ ise fonksiyon değişmediğinden dur.

Eğer $\|\Delta x\| \leq \varepsilon_2$ ise değişkenler değişmediğinden dur.

Eğer $\|\nabla f(\mathbf{x_{k+1}})\| \leq \varepsilon_3$ ise sonuca yakınsadığından dur.

Eğer k+1 = N ise iterasyon bittiğinden dur.

Adım 4: Eğer algoritma sonlandırılmamışsa

$y = \nabla f(\mathbf{x_{k+1}}) - \nabla f(\mathbf{x_k})$

$\Delta x = s_k \mathbf{p_k}$

$B = (yy^T) / (y^T / \Delta x)$

$C = (\nabla f(x_k)\, \nabla f(x_k)^T) / (\nabla f(x_k)^T \mathbf{p_k})$

$A_{k+1} = A_k + B + C$

$k \leftarrow k + 1$

Adım 2'ye git.

3.2.2.2 Davidon-Fletcher-Powell (DFP) Yöntemi

VMM ailesinin başka bir üyesidir. Karesel bir yakınsamaya sahiptir. DFP'de de metric matrisi kullanılır ve genellikle başlangıç olarak birim matris tercih edilir. Çözüme ulaşıldığında ise metric matrisi Hessian matrisinin tersi olur. Algoritması aşağıdaki gibidir [3]:

Adım 1: Başlangıç noktası = $\mathbf{x_1}$, metric = $\mathbf{A}$, iterasyon sayısı=N belirle.

Sonlandırma kriterleri ε_1, ε_2, ε_3 değerleri belirle.

k=1 ile iterasyonu başlat.

Adım 2: $\mathbf{x_k}$ noktasındaki gradyant vektör = $\nabla f(\mathbf{x_k})$ hesapla.

$\mathbf{p_k} = -\mathbf{A_k} \nabla f(\mathbf{x_k})$

Bir boyutlu optimizasyon ile $f(\mathbf{x_k} + \mathbf{s p_k})$'yi minimum yapan $\mathbf{s_k}$'yi bul.

$\mathbf{x_{k+1}} = \mathbf{x_k} + \mathbf{s_k p_k}$

Adım 3: $\Delta f = f(\mathbf{x_{k+1}}) - f(\mathbf{x_k})$; $\Delta x = \mathbf{x_{k+1}} - \mathbf{x_k}$

Eğer $|\Delta f| \leq \varepsilon_1$ ise fonksiyon değişmediğinden dur.

Eğer $\|\Delta x\| \leq \varepsilon_2$ ise değişkenler değişmediğinden dur.

Eğer $\|\nabla f(\mathbf{x_{k+1}})\| \leq \varepsilon_3$ ise sonuca yakınsadığından dur.

Eğer $k+1 = N$ ise iterasyon bittiğinden dur.

Adım 4: Eğer algoritma sonlandırılmamışsa

$y = \nabla f(\mathbf{x_{k+1}}) - \nabla f(\mathbf{x_k})$

$z = A_k y$

$\Delta x = s_k \mathbf{p_k}$

$B = (\Delta x \Delta x^T) / (\Delta x / y^T)$

$C = (z z^T) / (y^T z)$

$A_{k+1} = A_k + B + C$

$k \leftarrow k + 1$

Adım 2'ye git.

4. SEZGİSEL PROGRAMLAMA

Sezgisel programlama algoritmaları, çoğunlukla optimizasyon olmak üzere, herhangi bir amacı gerçekleştirmek üzere doğal olguları kullanan algoritmalardır. Bu algoritmalar, çözüm uzayında optimum çözüme yakınsaması ispat edilemeyen algoritmalar olarak da adlandırılır. Bu tür algoritmalar kesin çözümü garanti edemezler ve sadece kesin çözüm yakınındaki bir çözümü garanti edebilirler. Kesin çözüm bulamadığı halde sezgisel algoritmaların tercih edilme nedenleri [1]:

i. Optimizasyon problemi, kesin çözüm bulma işleminin tanımlanamadığı bir yapıya sahip olabilir.
ii. Anlaşılabilirlik açısından sezgisel algoritmalar karar verici için daha basit olabilir.
iii. Sezgisel algoritmalar öğrenme amaçlı ve kesin çözümü bulma işleminin bir parçası olarak kullanılabilir.
iv. Bunların yanı sıra sezgisel algoritmalar esnek yapıya sahiptirler ve bu sayede bir çok probleme uyum sağlayabilirler.

Sezgisel algoritmalar bu çalışmamızda 3 farklı sınıfa ayrılmıştır: Evrimsel hesaplama algoritmaları, gelişim algoritmaları, sürü temelli algoritmalar.

4.1 Evrimsel Hesaplama Algoritmaları

Evrimsel hesaplama yöntemleri belli bir popülasyonla çözüme başlayan ve bu popülasyonu döngüsel olarak büyüten ve/veya geliştiren algoritmaları barındırır. Evrimsel yöntemlerin temel yapı taşları; seçme, üreme, mutasyon ve çaprazlamadır. Başlangıç popülasyonu her döngüde bu işlemlerin tamamına veya birkaçına tabi tutularak yeni popülasyonlar üretilir ve çözüme buradan devam edilir.

Evrimsel hesaplama algoritmalarının temel işlem adımları aşağıdaki gibidir:

i. Başlangıç popülasyonu yarat.

ii. Her birey için uygunluk değerlerini hesapla.
iii. Seçilmiş bireylerden yeni popülasyon yarat.
iv. Popülasyona çaprazlama, mutasyon vb gibi evrimsel hesaplama teknikleri uygula.
v. Durdurma kriterleri sağlanıncaya kadar adım ii'ye git.

Evrimsel yöntemlerin en bilineni genetik algoritmalardır. Öte yandan evrimsel stratejiler ve genetik programlama da gayet fazla kullanılan yöntemlerdir.

4.1.1 Genetik Algoritmalar

Genetik algoritmalar canlı DNA'sının kendini eşleme ve çoğalma mantığını problem çözmek için kullanan algoritmalardır. Canlı DNA'sı iki sıra gen dizisinden oluşur. Yeni gen üretimi esnasında bu ikililer ayrılır ve her birinin karşısına aynı özelliği taşıyan genler gelecek şekilde yeni gen dizileri üretilir. Bu üretim sırasında kimi çekinik genler karşılarına baskın gen geldiği için işlevsiz hale gelebilir. Bazı durumlarda mutasyon yaşanıp sıralamada değişiklikler olabilir.

Bütün bu işlemler genetik algoritmalarda benzetimlerle kullanılmaktadır. Problemin çözümüne çoğunlukla rastgele bir popülasyonla başlanır. Buradaki popülasyon tabiri çözüm kümeleri için kullanılmaktadır. Bu popülasyon daha sonra sırasıyla yeni birey üretimi, bu bireylerden kötü olanların elenmesi, geriye kalanlardan seçme, seçilenlerden mutasyon ve çaprazlama gibi işlemlere tabi tutulur. Sonuç olarak problem rastgele değerlerden başlayıp genetik süreçlerle daha iyi sonuçlara gitmektedir. Basit bir genetik algoritma aşağıdaki gibidir [1]:

Adım 1: Çözümlerin bir başlangıç popülasyonunu oluştur.
Adım 2: Popülasyondaki her çözümün uygunluk değerini hesapla.
Adım 3: Durdurma kriteri sağlanmıyorsa
Doğal seçim yap (uygunluk değeri daha iyi olan çözümleri sakla).
Çaprazlama yap (mevcut iki çözümden yeni iki yapı üret).
Mutasyon yap (çözümlerde rastgele değişmeler meydana getir).

Adım 4: **Adım 2**'ye git.

4.1.2 Diferansiyel Gelişim Algoritması

Özellikle sürekli verilerin söz konusu olduğu problemlerde etkin sonuçlar verebilen, işleyiş ve operatörleri itibariyle genetik algoritmaya dayanan, popülasyon tabanlı bir optimizasyon tekniğidir [5].

Diferansiyel gelişim algoritması tamamen düzenlenmiş uzayda ve gerçek değerli parametreler ile çalışır. Ayrık bir optimizasyon algoritması değil, özellikle nümerik optimizasyon için geliştirilmiş bir gelişim algoritmasıdır. Algoritmanın temel adımları aşağıdaki gibidir [1]:

Adım 1: Kontrol parametrelerinin
(D, Gmax, NP $\geq$ 4, F $\in$ (0, 1+), CR $\in$ [0, 1]) değerlerini ve parametre sınırlarını x^{lo} ve x^{hi} ata.

Adım 2: Başlangıç popülasyonunu oluştur.
$\forall i \leq NP \wedge \forall j \leq D : x_{j,i,G=0} = x_j^{lo} + rand_j[0, 1]*(x_j^{hi}-x_j^{lo})$
$i = (1,2,...,NP)$, $j = (1,2,...,NP)$, $G = 0$, $rand_j[0, 1] \in [0, 1]$

Adım 3: Durdurma kriteri sağlanıncaya kadar aşağıdakileri yap.
Mutasyon
Çaprazlama
$r_1, r_2, r_3 \in \{1,2,...,NP\}$, $r_1 \neq r_2 \neq r_3 \neq i$ (rastgele seçilmiş)
$j_{rand} \in \{1,2,...,NP\}$, (rastgele seçilmiş)
Eğer randj[0,1) < CR v j = jrand
$\forall j \leq D, u_{j,i,G+1} = x_{j,r3,G} + F * (x_{j,r1,G} - x_{j,r2,G})$
Değilse
$\forall j \leq D, u_{j,i,G+1} = x_{j,i,G}$

Adım 4: Seçme
Eğer $f(u_{i,G+1}) \leq f(x_{i,G})$
$x_{i,G+1} = u_{i,G+1}$
Değilse
$x_{i,G+1} = x_{i,G}$

4.2 Gelişim Algoritmaları

Gelişim algoritmaları başladığı noktadan belirli kurallar çerçevesinde adım adım çözüme giden algoritmalardır. Evrimsel algoritmalardan en büyük farkı mutasyon gibi etkilerle çözüm uzayında alakasız bir noktadan başlama seçeneği yoktur. İlk belirlenen noktalardan yola çıkarak bu noktaları geliştirerek ilerlerler. En bilinen örnekleri arasında tabu araştırma algoritması, ısıl işlem algoritmaları ve diferansiyel gelişim algoritması bulunmaktadır.

4.2.1 Tabu Araştırma Algoritmaları

Tabu araştırması, başlangıçta tümleşik optimizasyon problemleri için geliştirilmiş sezgisel yaklaşımlardan biridir [6].

Gelişim algoritmalarının en büyük sorunu bir sonraki adımın uygunluğunun kötü olması ve bu adımdan yola devam edilerek daha kötü sonuçlara varılmasıdır. Daha önceki adımlarda bulunan kötü sonuçlardan kurtulabilmek için tabu listeleri oluşturulan algoritma tiplerine adından anlaşılacağı gibi tabu araştırma algoritmaları denir. Bu tip algoritmaların en büyük eksiği bulunan çözümün o an için kötü olup da daha sonraki adımlarda iyileşebilme ihtimalini değerlendirememesidir. Tabu araştırma algoritmaları yerel çözümleri bulan algoritmalardır fakat bazı geliştirmelerle ve/veya eklentilerle küresel çözümlere ulaşan çeşitleri de vardır. Aşağıda temel algoritma verilmiştir [1]:

Adım 1: Bir başlangıç çözüm (S) al. Başlangıçta değer atanması gereken parametrelerin atamasını yap.

Adım 2: Komşu çözümler üret ve bu çözümler arasından en iyi kabul edilebilir olanı (S_{eniyi}) seç. Kötü çözümleri tabu listesine ekle.

Adım 3: Mevcut çözümü (S), S_{eniyi} ile yer değiştir ve tabu listesini yenile.

Adım 4: Durdurma kriteri sağlanıncaya kadar Adım 2 ve Adım 3'ü tekrar et.

4.2.2 Isıl İşlem Algoritması

Metal malzemelerde katı halde sıcaklık değişimleri ile bir ya da birbirine bağlı birkaç işlemle amaca uygun özellik değişmelerinin sağlanması olayına ısıl işlem denir [1]. Isıl işlem uygulanan metal ısıtılarak şekillendirilir, bu sıcaklıkta belli bir süre tutulur ve ardından soğutulur. Isıl işlem algoritmaları da bu süreçlere benzer ısıtma, bekleme ve soğutma adımlarından oluşur.

Isıl işlem algoritmaları başlangıç noktasından yeni komşulara gider. Tabu araştırmasından farklı olarak kötü sonuçları da çözüm sürecine dahil eder. Bütün komşulara olasılık değerleri atanarak bir sonraki çözüme gidilir. Bu sayede başlangıçta bir çözüm kötü de olsa ondan sonraki adımda daha iyi sonuca gidebilme ihtimali değerlendirilmiş olur. Burada olasılık değeri, T sıcaklığında enerjide δE genlikli bir artışın olma olasılığıdır. Denklem 4.1'de gösterilmektedir.

$$p(\delta E) = \exp(-\delta E/kT) \quad (k=\text{Boltzman sabiti}) \tag{4.1}$$

Isıl işlemlerin algoritması aşağıdaki gibidir [1]:

Adım 1: Rastgele başlangıç çözümü (S) üret. Sıcaklık (T) için başlangıç değeri (T_s) belirle ve diğer parametreleri tayin et.

Adım 2: Komşu bir çözüm $S' \in N(S)$ üret ve bu üretilen çözümle S çözümünün amaç değerleri (C(S) ve C(S')) arasındaki farkı $\Delta=C(S') - C(S)$ hesapla.

Adım 3: Eğer $\Delta<0$ veya ($\Delta>0$ ve T'de ($- \Delta/e^T > \Theta$) rastgelelik işlemi Denklem 4.1 ile kabul edilmiş ise

S çözümünü S' ile yer değiştir.

(Θ, $0<\Theta<1$ olacak şekilde rastgele bir sayı)

Adım 4: (i) Kullanılan soğutma tarifesine göre,

(ii) Üçüncü adımda üretilen çözümlerde gelişme olup olmamasına göre,

(iii) S çözümüne ait komşuluğun tamamen araştırılıp araştırılmamasına göre,

(iv) Varsa başka kriterlere göre,

T sıcaklığını değiştir.

Adım 5: Durdurma kriteri sağlanana kadar **Adım 2**'ye git.

4.3 Sürü Temelli Algoritmalar

Kalabalık halde yaşayan, birbirleriyle etkileşen, dağınık yapılı, organize topluluğa sürü denir. Sürü halinde yaşayan hayvanlarda her bireyin kendine ait ama bütün sürünün geleceğini etkileyebilecek görevleri vardır. Bu sayede amaçlanan durum her bireyin katkısıyla çok daha çabuk halde gerçekleştirilebilir.

Sürü temelli algoritmalar genelde böcek sürüleri olmak üzere birçok hayvan topluluğunun yem bulma ve/veya tehlikelerden kaçınma alışkanlıklarını modelleyen algoritmalardır. Bu tip algoritmalarda probleme, sürü elemanı kadar üretilen rastgele ve/veya belli çözüm noktalarıyla amaç fonksiyonunun dağınık olarak farklı bölgelerinden başlanır. En bilinen örnekleri karınca kolonisi, arı kolonisi ve parçacık sürü optimizasyonu yöntemleridir.

4.3.1 Karınca Kolonisi Algoritması

Karınca kolonisi optimizasyonu karıncalar, termitler ve diğer sosyal böceklerin sürü zekaları üzerine çalışan bir alandır [9]. Karınca kolonisi optimizasyonu meta sezgiselinde yapay karıncaların kombinatoryal problem için oluşturulan ağ yapısı üzerinde doğal karıncaların yiyecek arama davranışlarını taklit ederek en kısa yolu bulması amaçlanmaktadır [8]. Karıncalar yem bulmak için farklı yönlere sürekli birey gönderirler. Her eleman hareket halindeyken arkasında feromon adı verilen bir çeşit koku bırakır. Eğer bulunan yem kaynağı kaynaklar arasında daha elverişliyse diğer bireyler de bu kaynağa yönelir. Bu sayede tercih edilen kaynak yolu üzerinde biriken feromon miktarı artar. Bir süre sonra sürünün bütün elemanları daha fazla feromon olan bu yola yönelir.

Bütün sezgisel yöntemlerde olduğu gibi bu yöntem de kesin sonucu garanti edemez. Bunun nedeni; karınca sürülerinin belli durumlardan dolayı yakın olan yem kaynaklarını gözden kaçırmış olma ihtimalinin var olmasıdır. Bu durum algoritmaya da yansımıştır. Basit bir karınca kolonisi algoritması aşağıdaki gibidir [1]:

Adım 1: Parametrelere ve hatların feromon miktarlarına başlangıç değerlerini ata ve sayacı sıfırla.

Adım 2: Aşağıdaki adımları durdurma kriterleri sağlanıncaya kadar tekrarla.

Adım 3: Tüm karıncalar için feromon maddesine bağlı olarak yollar üret.

Adım 4: Yolların uzunluklarını hesapla.

Adım 5: Yolların uzunluklarına bağlı olarak mevcut feromon miktarını güncelle.

Adım 6: Şu ana kadar bulunan en kısa yolu hafızada tut.

4.3.2 Yapay Arı Kolonisi Algoritması

Doğal bir arı kolonisinde arılar arasında yapılacak işlere göre bir görev paylaşımı vardır. Bu paylaşım merkezi bir birim olmadan kendi kendilerine gerçekleşmektedir. Bu iş paylaşımı ve kendi kendine organize olabilme sürü zekasının iki önemli özelliğidir [9].

Arılar yiyecek kaynağı bulduğu zaman kaynağın yerini diğer arılara göstermek için bir çeşit dans yaparlar. Görevleri sadece kaynak bulmak olan bir grup arı; kaynak bulduğu zaman kaynağın yerine, güneşin açısına, kaynağın besin değerlerine vb bağlı olarak görevleri sadece bal toplamak olan arılara haber verir. Toplayıcı arılar da bu danstan aldıkları bilgilere göre çiçeklere giderken yükseklik, enerji tüketimi, yük taşıma miktarı vb gibi ayarlamaları yapar. Bir kaynak tükenirse yeni kaynak arayışı başlar ve yeni konuma göre yeni bir dans yapılır.

Arı kolonisi algoritması bal arılarının kovanları yakınındaki kaynakları bulmak için geliştirdikleri zeki davranışların benzetimini yapan algoritmadır [6]. Algoritmanın en temel hali aşağıdaki gibidir [1]:

Adım 1: Başlangıç yiyecek kaynağı bölgelerini üret.

Adım 2: Durdurma kriterleri sağlanıncaya kadar aşağıdaki adımları tekrarla.

Adım 3: Görevli arıları yiyecek kaynağı bölgelerine gönder.

Adım 4: Olasılıksal seçmede kullanılacak olasılık değerlerini görevli arılardan gelen bilgiye göre hesapla.

Adım 5: Gözcü arılara olasılık değerlerine göre yiyecek kaynağı bölgesi seçtir.

Adım 6: Bırakılacak kaynakları bıraktır ve kaşif arı üret.

4.3.3 Parçacık Sürü Optimizasyonu (PSO) Algoritması

Parçacık Sürü Optimizasyonu algoritması 1995 yılında Kennedy ve Eberhart tarafından doğrusal olmayan, sınırlı ve/veya sınırsız, çok modelli fonksiyonların çözümü için geliştirilmiş bir tekniktir [11, 12]. Algoritma, kuş ve balık sürülerinin sosyal davranışlarının benzetimidir [11-13].

PSO az parametre ile çalışan bir algoritmadır. Diğer sezgisellerden farklı olarak sadece her parçacığın hız ve konum bilgilerinin tutulması yeterlidir. Bu kadar az parametre olması da diğerlerine göre daha hızlı sonuç vermesine yardımcı olmaktadır.

PSO çalışma prensibi sayesinde yerel optimumlara takılmaktan kurtulur. Birçok parçacık ile çözüme başlanır ve her turda konum bilgileri güncellenir. Konumu çözüme daha yakın olan parçacığa göre belirlenen hız vektörü ile diğer parçacıkların konumları güncellenir. Bu sayede yerel optimuma yakalanan parçacık olsa bile daha iyi sonuç bulan parçacıklar sayesinde bu noktadan kurtulurlar.

4.3.3.1 Parçacık Sürü Optimizasyonu Kavramları

PSO terminolojisi aşağıdaki gibidir:

c_1 : parçacık hızlandırma faktörü

c_2 : sürü hızlandırma faktörü

r_1, r_2 : [0, 1] aralığında rastgele üretilen sayılar

w : atalet ağırlığı

$Pbest_i$: i parçacığının yerel en iyi değeri

G_{best}	: sürünün en iyi değeri
$X_i(k)$	: i parçacığının k anındaki konum bilgisi
$V_i(k)$	: i parçacığının k anındaki hız bilgisi
PS	: Parçacık sayısı
N	: İterasyon sayısı

PSO temelinde sadece 2 tane vektör denklemi barındırır. Bunlardan biri konum, diğeri ise hızdır. Konum vektörünü hesaplarken hız vektörünün bilgisinden faydalanılmaktadır. Dolayısıyla hızın yönü konumuzda farklılıklar yaratacaktır.

Konum vektörü hesaplanırken Denklem 4.2'de de belirtildiği gibi hız vektörü kullanılmaktadır.

$$X_i(k+1) = X_i(k) + V_i(k+1) \tag{4.2}$$

Hız vektörü, 1995 yılında Eberhart ve Kennedy'nin ilk çalışmalarında, sapmalara, hatalara ve performans kayıplarına neden olabiliyordu. Bu haliyle ortaya atılan denklem aşağıdaki gibidir [11]:

$$V_i(k+1) = V_i(k)+ c_1*r_1(k)*(Pbest_i(k) - X_i(k)) + c2*r2(k)*(G_{best} (k) - X_i(k)) \tag{4.3}$$

Denklem 4.3'ün verdiği hatalı sonuçlardan sonra, 1998 yılında yayımlanan Eberhart ve Shi'nin çalışmasında [14], atalet ağırlığı eklenmiş ve daha kararlı sonuçlar elde edilmiştir. Atalet ağırlığı eklenmiş hali Denklem 4.4'te verilmiştir:

$$V_i(k+1)=w*V_i(k)+ c_1*r_1(k)*(Pbest_i(k) - X_i(k)) + c_2*r_2(k)*(G_{best} (k) - X_i(k)) \tag{4.4}$$

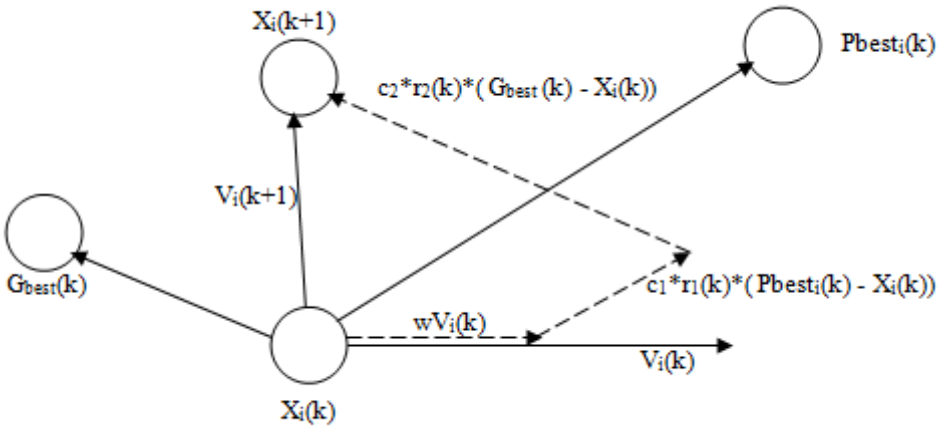
Hız vektörünün denklemi Denklem 4.4 haline getirildikten sonra bulunması gereken parametremiz atalet ağırlığıdır. Atalet ağırlığının değerini belirleme konusunda bir çok çalışma bulunmaktadır. Aşağıda yöntemlerden birkaçı bulunmaktadır [12]:

$$w_1 = (w_1 - w_2) * \{(ToplamTurSayısı - OAnkiTur)/TurSayısı\} + w_2 \tag{4.5}$$

$$w(i) = w_0 * \exp(OAnkiTur/ToplamTurSayisi)^n \tag{4.6}$$

$$w = 0.5 + (rand(.) / 2) \tag{4.7}$$

Bu çalışmada atalet ağırlığı olarak Denklem 4.7 kullanılmıştır. [0, 1] arası rastgele üretilmiş bir sayının yarısını 0.5 ile toplayarak atalet ağırlığını buluyoruz. Denklem gereği bulunan sonuç en fazla 1, en az 0.5 olmaktadır. Şekil 4.1'de hız ve konum vektörüne göre parçacığın izlediği yol görülmektedir [15, 16].



Şekil 4.1: Parçacık Hareketi

Algoritmada kullanılacak gerekli denklem tanımlamaları yukarıda yapılmıştır. Aşağıda PSO algoritması verilmektedir [17, 18]:

Adım 1: Parçacık sayısını (PS) belirle ve her parçacık için ilk konum (X), ilk hız (V) değerlerini ata. Algoritma parametreleri olan c_1, c_2, r_1, r_2, w ve N ata. Her parçacık için P_{best} değerini ilk konumlar olarak ata. En iyi P_{best}'i G_{best} olarak ata. Sonlandırmak için istenilen başka kriter varsa ata.
k=1 ile başla.

Adım 2: k <= N olana veya sonlandırma kriteri sağlanıncaya kadar devam et.

Adım 3: Her parçacık için uygunluk değerlerini hesapla.
Her parçacık için yeni çıkan değerler eski değerlerden iyiyse bu değerleri P_{best} yap.
Tüm P_{best} arasında en iyi konumu G_{best} olarak belirle.

Adım 4: Her parçacık için Denklem 4.4'ü kullanarak yeni hızları belirle.

Her parçacık için Denklem 4.2'yi kullanarak yeni konumları belirle.

$k \leftarrow k + 1$

Adım 2'ye git.

Literatürde PSO üzerinde yapılan çalışmalar sonucu parçacık sayısının (PS) 20, 30 veya 40, hızlandırma faktörleri olan c_1 ve c_2'nin değerlerinin 2 ve iterasyon sayısının (N) da 1000, 2000 veya 3000 olması halinde en doğru sonuçlar bulunacağı tespit edilmiştir. Bu değerler uygulamanın karmaşıklığına göre farklılıklar gösterebilmektedir.

5. GELİŞTİRİLEN YÖNTEM

Daha önceden de belirtildiği gibi bu çalışmanın amacı, kesin çözüm bulan ve görece daha hızlı bir teknik geliştirmektir. Bunu sağlamak için numerik yöntemlerin kesin çözüm bulma yeteneği sezgisel yöntemlerin hızlı sonuç alma yeteneği ile birleştirilmiştir.

5.1 Açıklamalar

Geliştirilen teknik iki farklı alandaki optimizasyon yönteminin birleştirilmesi ile oluşturulan melez bir yapıdır: Numerik bir yöntemle sezgisel bir yöntemin bir arada çalışması.

Numerik yöntemlerin kötü yanlarından biri yerel minimum noktalara takılma sorunudur. Öte yandan yerel de olsa kesin minimum noktayı garanti edebilmektedirler. Sezgisel yöntemler ise kesin noktayı garanti edememekle birlikte küresel çalışmaktadır.

Bu çalışmada rastgele bir x noktasından yola çıkarak, numerik bir yöntem ile yerel minimum olan y_{min} noktasına ulaşılmaktadır. Daha sonra bulunan bu nokta küresel minimum varsayılarak PSO çalıştırılmaktadır. Böylece fonksiyondaki tüm y_{min} noktaları bulunmaktadır. Bulunan bu noktalardan herhangi biri ile tekrar numerik yöntem çalıştırılmakta, ardından tekrar PSO'ya devretmektedir. Bu döngü ya minimuma ya da tur sınırına ulaşana kadar devam etmektedir.

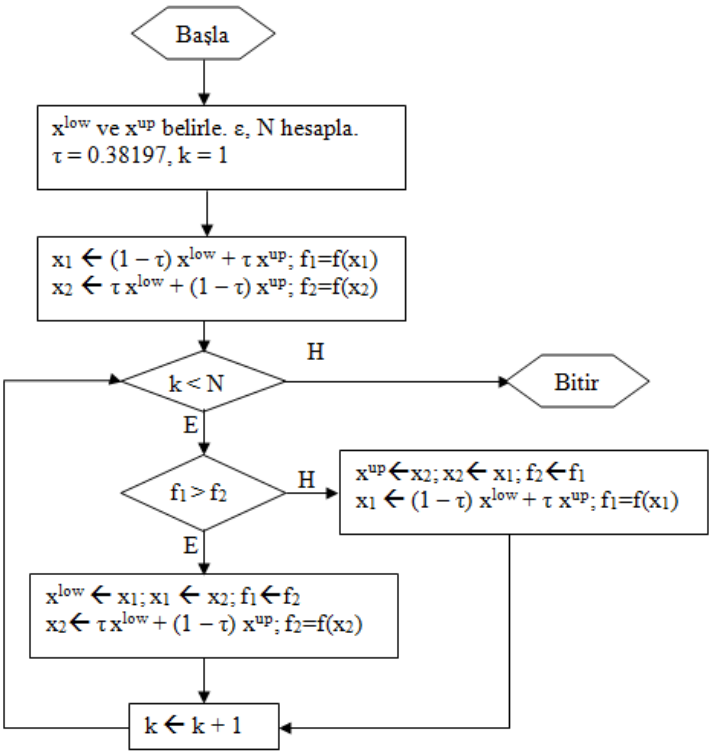
Ayrıca geliştirilen algoritmanın tersi mantıkla işleyen bir yöntemle karşılaştırma yapılıştır. Bu yönteme göre PSO problemde minimuma en yakın noktaya ulaşılmakta ve oradan da kesin çözüm için BFGS ile düzeltme yapılmaktadır.

5.2 Akış Şeması

Bu çalışmada kullanılan yöntemler, kesin çözüm bulabilme yeteneklerinden dolayı numerik yöntemlerden BFGS ve hızlı sonuçlar alarak kesin çözüme yakınsamalarından dolayı sezgisel yöntemlerden PSO'dur. Numerik yöntemler geliştirilen yöntem gereği sezgiselle birlikte çalışıyor. Ayrıca BFGS çok boyutlu numerik optimizasyon tekniği olduğu için ve adım aralığının bulunması gerektiği için ABY kullanılmaktadır. Aşağıda sırasıyla her yöntemin akış diyagramı verilmiştir. Ardında da bu algoritmaların nasıl birleştiğini gösteren akış diyagramı bulunmaktadır.

5.2.1 Altın Bölme Yöntemi Akış Diyagramı

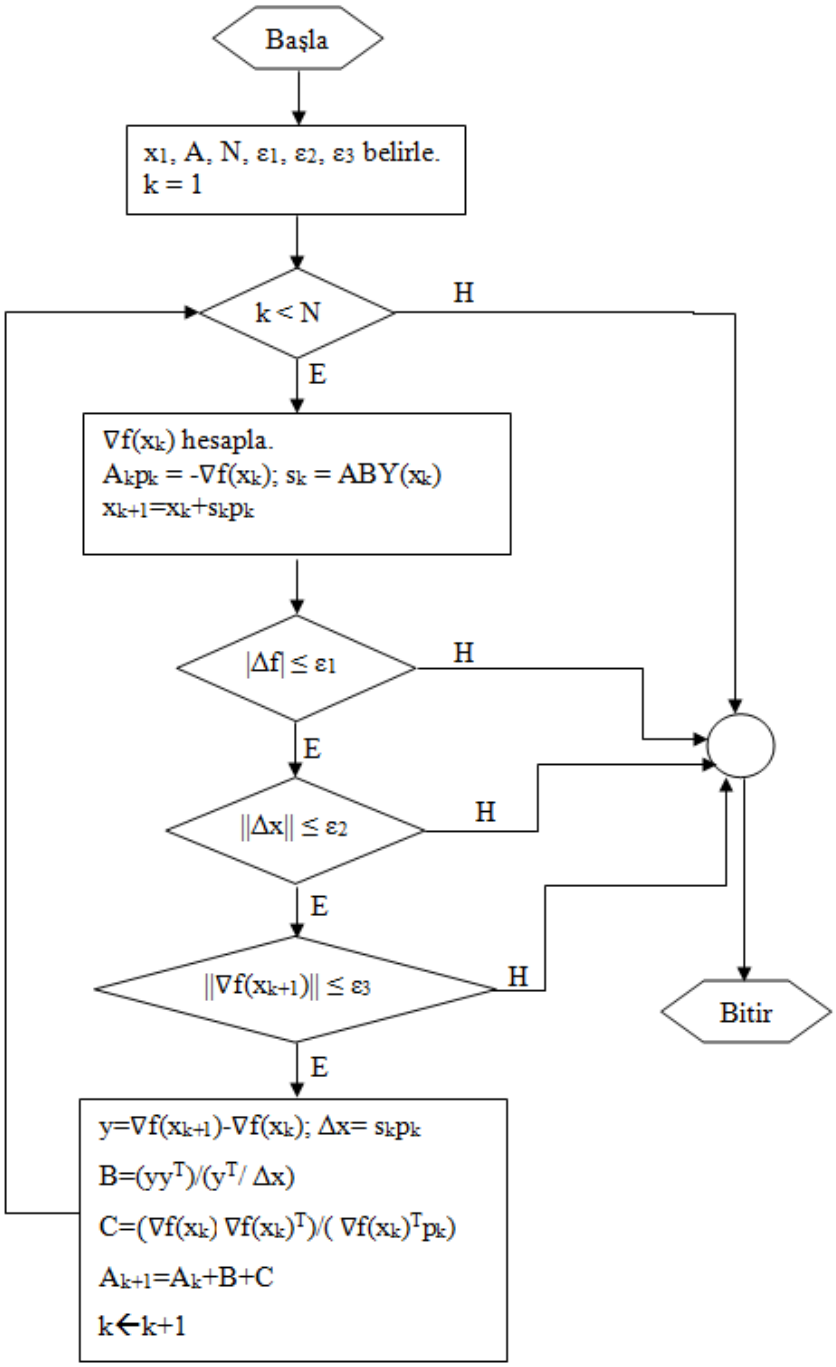
Bölüm 3.1.2.1'de algoritması verilmiş olan altın bölme yönteminin akış diyagramı Şekil 5.1'de verilmiştir. ABY daha sonra verilen akış diyagramlarının içerisinde kullanılmaktadır. Gösterim şekli olarak ABY(x) notasyonu kullanılmıştır.



Şekil 5.1: BY Akış Diyagramı

5.2.2 Broydon-Fletcher-Goldfarb-Shanno Yöntemi Akış Diyagramı

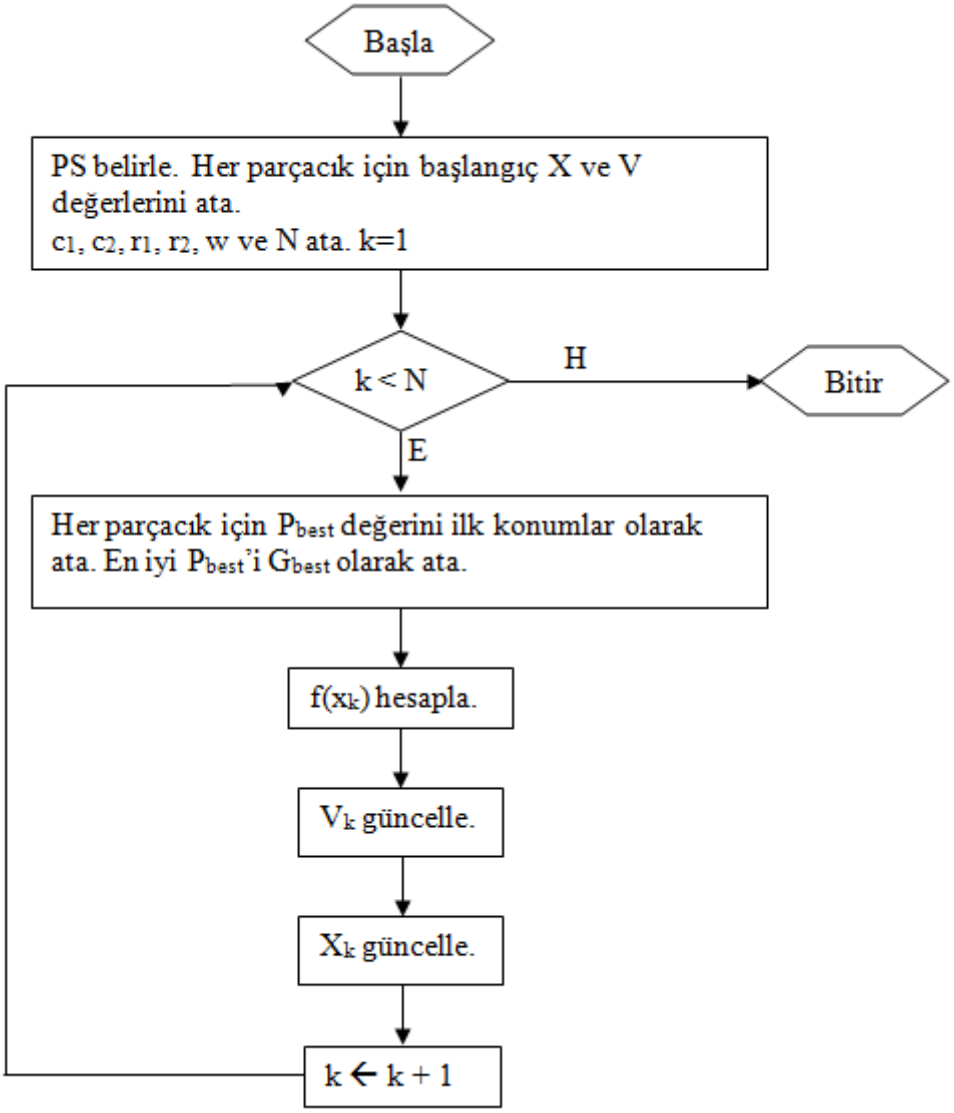
Tez çalışmamızda BFGS yöntemi PSO ile birlikte çalışan numerik yöntemlerden biridir. Bölüm 3.2.2.1'de verilen algoritmasının akış diyagramı Şekil 5.2'deki gibidir.



Şekil 5.2: BFGS Akış Diyagramı

5.2.3 Parçacık Sürü Optimizasyonu Akış Diyagramı

PSO algoritmasının, bölüm 4.3.3.1'deki algoritmaya göre, akış diyagramı Şekil 5.3'te verilmiştir.

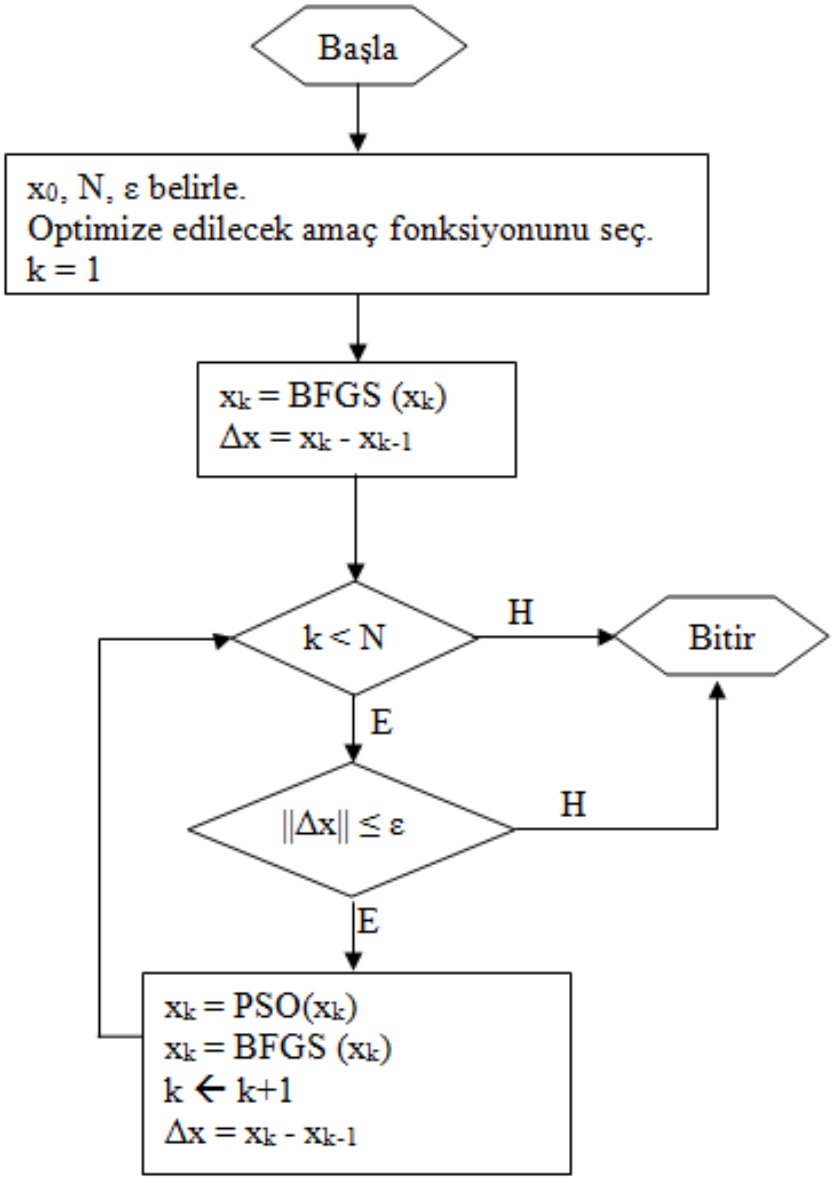


Şekil 5.3: PSO Akış Diyagramı

5.2.4 Birleştirilmiş Akış Diyagramı

Yukarıda bahsedilen yöntemlerin çalışmamızdaki kullanımı Şekil 5.4'teki akış diyagramı ile gösterilmiştir. Akış diyagramından da anlaşıldığı gibi bir başlangıç x noktasından yola çıkılarak seçilen amaç fonksiyonunu, seçilen numerik yönteme

göre, gradyant yöntem ve PSO arasında sırayla optimize edilmektedir. Önceki x değeri ile bulunan x değeri arasında farkın en küçük olduğu ana kadar algoritma çalışmaktadır.



Şekil 5.4: Birleştirilmiş Akış Diyagramı

5.3 Algoritma

Yukarıda verilen birleştirilmiş akış diyagramı doğrultusunda çalışmanın algoritması aşağıda verilmiştir.

Adım 1: Bir başlangıç değeri x_0, durdurma kriteri ε, iterasyon sayısı N belirle.

Optimize edilecek amaç fonksiyonunu seç.

$x_k = BFGS(x_k)$

$k = 1$

Adım 2: $||\Delta x|| = norm(x_k - x_{k-1})$

Eğer $||\Delta x|| < \varepsilon$ ise değişkenler değişmediği için dur.

$k + 1 = N$ ise iterasyon bittiği için dur.

Adım 3: $x_k = PSO\ (x_k)$

$x_k = BFGS(x_k)$

$k \leftarrow k+1$

Adım 2'ye git.

Burada amaç fonksiyonu olarak Bölüm 6.1'deki test fonksiyonları kullanılmıştır. x_0 başlangıç noktası seçilen test fonksiyonunun araştırma bölgesi içinde üretilmiş rastgele bir noktadır. ε durdurma değeri olarak 10^{-3} tercih edilmiştir. Yapılan testlerde algoritmanın çözüme ulaşması için 2 veya 3 iterasyon yeterken biz uygulamamızda 20 iterasyon kullandık.

5.4 Karşılaştırma İçin Oluşturulan Algoritma

Karşılaştırma algoritması için kullanılan yöntem literatürde sık kullanılan bir yöntemlerdendir. Bu algoritmalarda genellikle sezgisel bir yöntem ile çözüm uzayında global minimuma en yakın nokta bulunur. Daha sonra problemin tipine göre uygun bir numerik yöntem veya tabu araştırma algoritması gibi yerel çözüm bulan bir başka sezgisel ile kesin çözüme ulaşılır. Bu tip algoritmaların genel adımları aşağıdaki gibidir:

Adım 1: Bir başlangıç değeri x_0, durdurma kriteri ε, iterasyon sayısı N belirle.

Kullanılacak sezgisel yöntemi (SY) ve numerik yöntemi (NY) belirle.

Optimize edilecek amaç fonksiyonun seç.

$k = 1$

Adım 2: $k < N$ ve ε şartı sağlanmamış ise

$x_k = SY(x_k)$

$x_k = NY(x_k)$

$k = k + 1$

Adım 2'ye git.

Geliştirdiğimiz algoritmanın karşılaştırılabilmesi için en uygun yöntemler olarak, karşılaştırma algoritmasında, sezgisel yöntem olarak PSO, numerik yöntem olarak da BFGS kullanılmıştır.

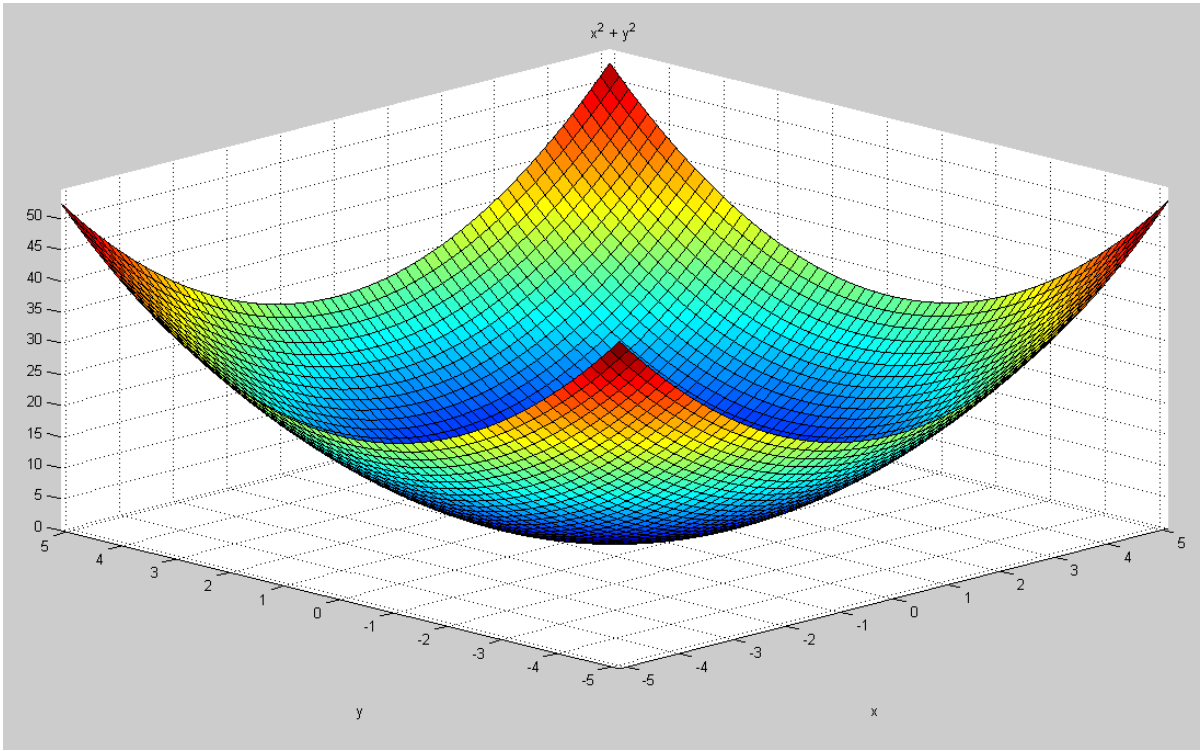
6. UYGULAMALAR

Geliştirilen algoritmayı test etmek için literatürde sıklıkla çalışılan test fonksiyonları kullanılmıştır. Numerik yöntemleri kullanabilmek için test fonksiyonlarının kısmi türevleri de alınmıştır.

6.1 Kullanılan Test Fonksiyonları

1) De Jong Fonksiyonu [4]: $f(x) = \sum_{i=1}^{n} x_i^2$

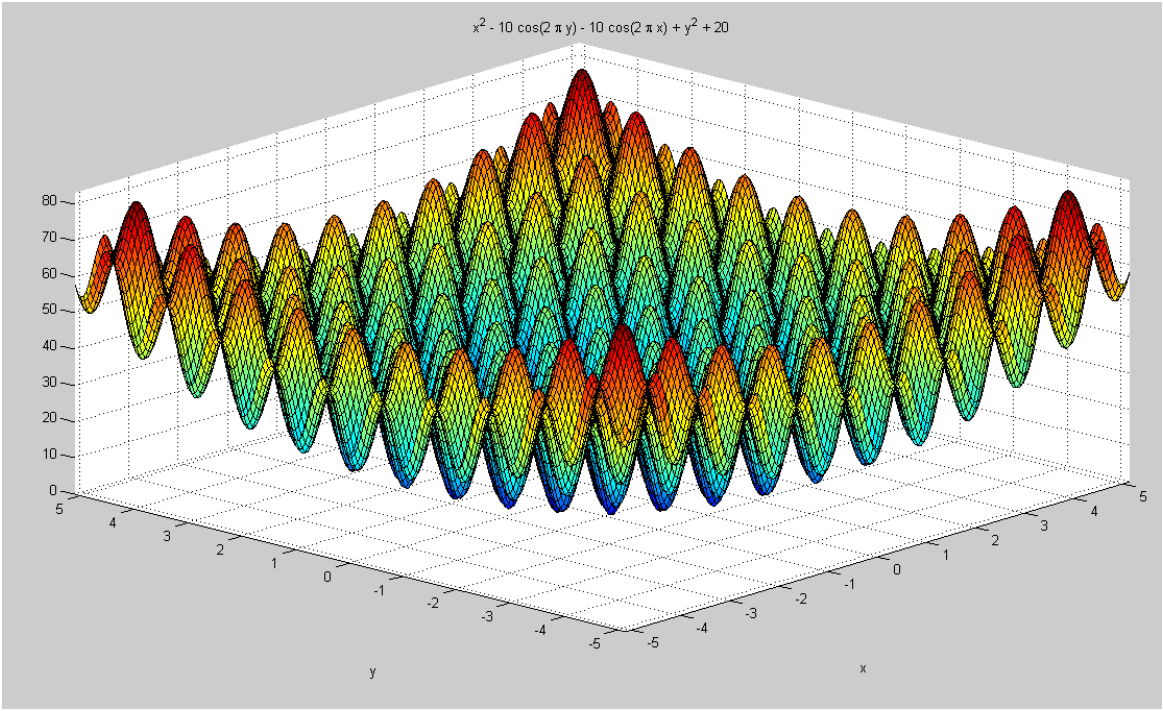
   En basit test fonksiyonudur. Fonksiyon sürekli, dışbükey ve tek modellidir. Test bölgesi genellikle $-5.12 \leq x_i \leq 5.12$, i=1, ..., n sınırları arasında aranır. Küresel minimum $x_i=0$, i=1, ..., n için f(x)=0 noktasında bulunmaktadır.



Şekil 6.1: De Jong Fonksiyonu

2) Rastrigin Fonksiyonu [4]: $f(x) = 10n + \sum_{i=1}^{n}(x_i - \cos(2\pi x_i)$

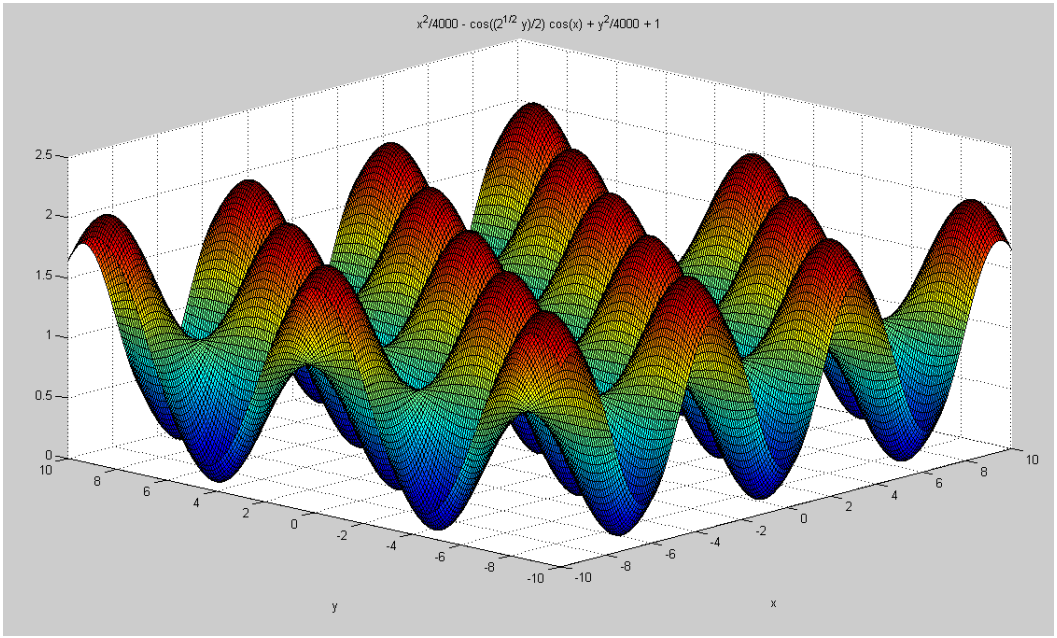
Çok modelli ve minimum noktaları düzenli dağılmış bir fonksiyondur. Test bölgesi genellikle $-5.12 \leq x_i \leq 5.12$, i=1, ..., n sınırları arasında aranır. Küresel minimum $x_i=0$, i=1, ..., n için f(x)=0 noktasında bulunmaktadır.



Şekil 6.2: Rastrigin Fonksiyonu

3) Griewangk Fonksiyonu [4]: $f(x) = \frac{1}{4000} + \sum_{i=1}^{n} x_i^2 - \prod_{i=1}^{n} \cos\frac{x_i}{\sqrt{i}} + 1$

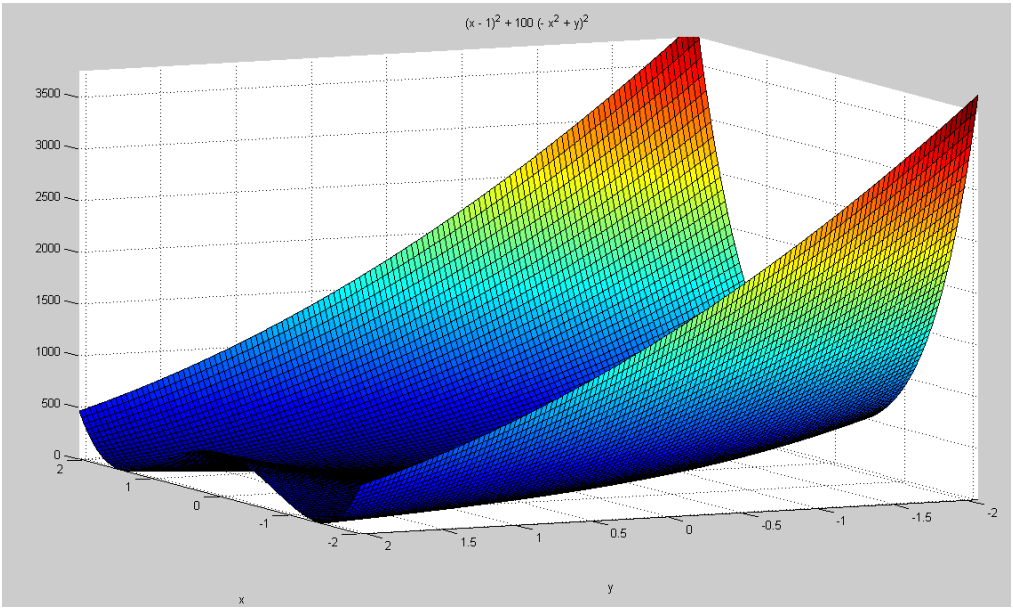
Rastrigin fonksiyonuna benzer bir yapıda ama daha geniş dağılmış minumum noktalara sahip çok modelli bir fonksiyondur. Test bölgesi genellikle $-600 \leq x_i \leq 600$, i=1, ..., n sınırları arasında aranır. Küresel minimum $x_i=0$, i=1, ..., n için f(x)=0 noktasında bulunmaktadır.



Şekil 6.3: Griewangk Fonksiyonu

4) Rosenbrock Fonksiyonu [4]: $f(x) = \sum_{i=1}^{n-1}(100((x_{i+1} - x_i^2)^2 + (1 - x_i)^2))$

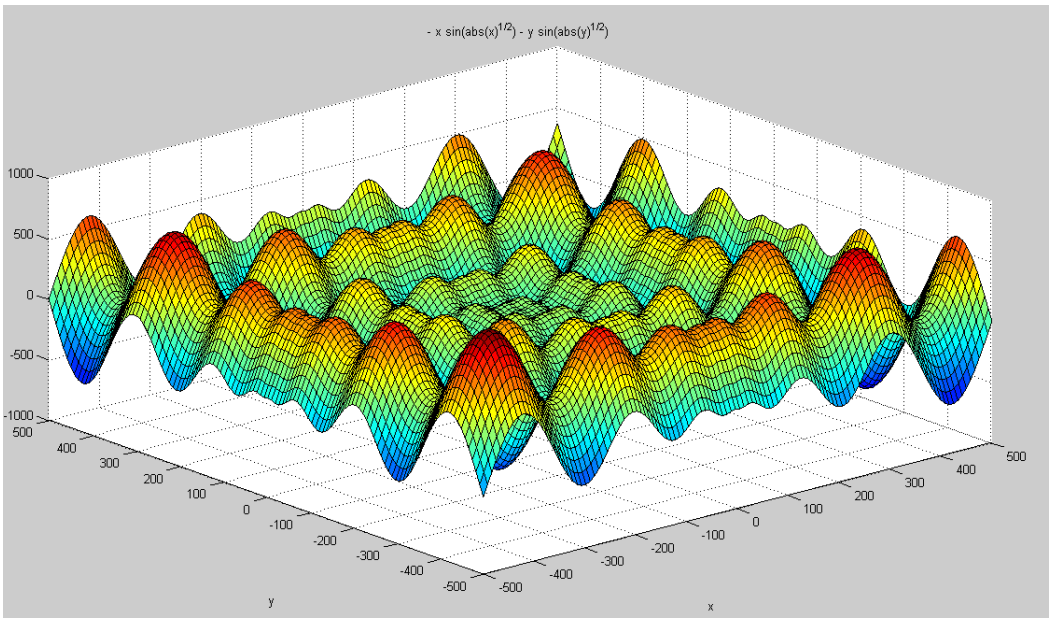
Rosenbrock vadisi veya muz fonksiyonu olarak da anılır. Küresel optimum uzun, geniş, parabolik şekilli vadide yatar. Test bölgesi genellikle $-2.048 \leq x_i \leq 2.048$, i=1, ..., n sınırları arasında aranır. Küresel minimum $x_i=1$, i=1, ..., n için f(x)=0 noktasında bulunmaktadır.



Şekil 6.4: Rosenbrock Fonksiyonu

5) Schwefel Fonksiyonu [4]: $f(x) = \sum_{i=1}^{n} \left(-x_i \sin\sqrt{|x_i|}\right)$

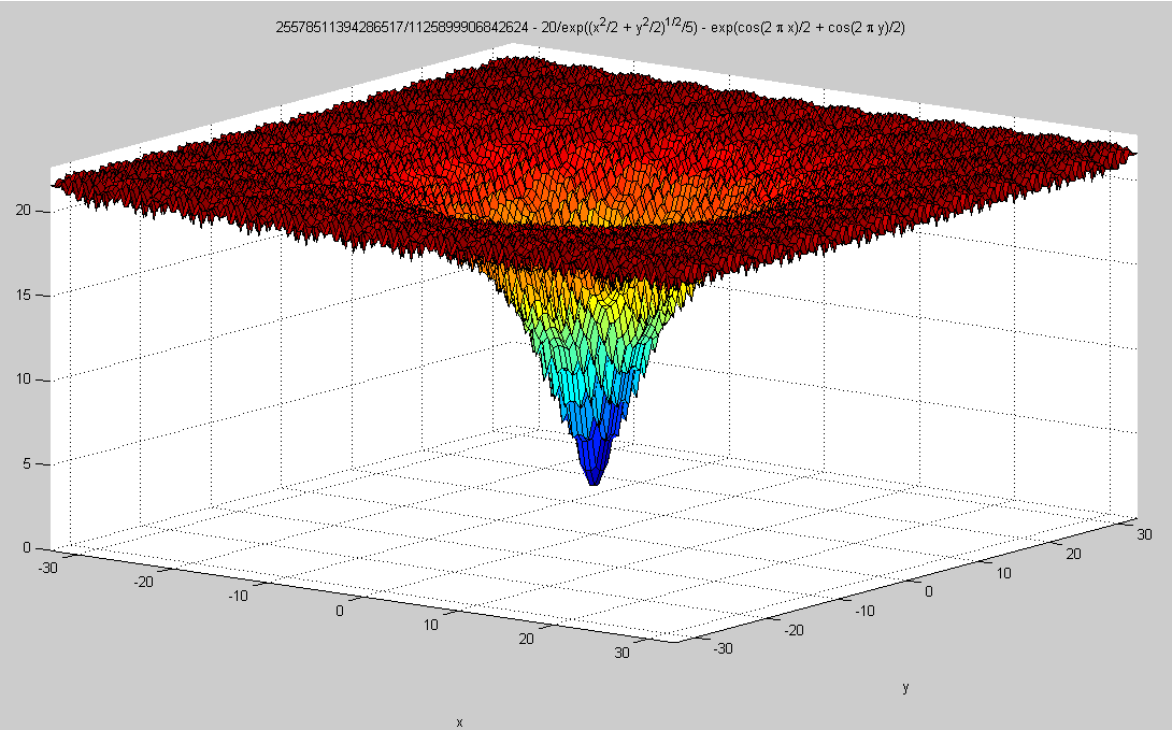
Test bölgesi genellikle $-500 \leq x_i \leq 500$, i=1, ..., n sınırları arasında aranır. Küresel minimum x_i=420.9687, i=1, ..., n için f(x)= - 418.9829 noktasında bulunmaktadır.



Şekil 6.5: Schwefel Fonksiyonu

6) Ackley Fonksiyonu [4]: $f(x) = -a \exp\left(-b\sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2}\right) - \exp\left(\frac{1}{n}\sum_{i=1}^{n} cos(cx_i)\right) + a + \exp(1)$

En çok kullanılanlardan biri olan çok modelli bir test fonksiyonudur. Genellikle a=20, b=0.2 ve c=2 π olarak önerilmektedir. Test bölgesi $-32.768 \leq x_i \leq 32.768$, i=1, ..., n sınırları arasında aranır. Küresel minimum x_i=0, i=1, ..., n için f(x)=0 noktasında bulunmaktadır.



Şekil 6.6: Ackley Fonksiyonu

6.2 Sonuçlar

Önerilen algoritma ile karşılaştırma algoritması 64 bit Windows 7 işletim sistemi ve Matlab R2011b(7.13.0.564) programlama ortamında geliştirilmiştir. Kullanılan sistem Intel Core i5-2410M işlemciye sahiptir.

Çalışmamızda fonksiyonlar 3 boyutlu ortamda test edilmiştir (f(x, y) = z). Algoritmaların sonuçlarının doğruluğunun kontrolü için Rastrigin, Griewangk ve Rosenbrock fonksiyonları ellişer defa çalıştırılmıştır, rastgele oluşturulan başlangıç değerleri iki algoritma için kullanılmıştır. Bu sayede aynı noktadan çözüme başlanılmış ve buldukları sonuçlar ve hızları karşılaştırılabilmiştir. Matlab programlama ortamının varsayılan olarak belirlediği hassasiyet 10^{-15} olup daha küçük değerler için 0 sonucunu vermektedir. Çalışmamızda varsayılan değer değiştirilmemiştir.

Tablo 6.1'de Rastrigin fonksiyonu için rastgele oluşturulmuş başlangıç değerleri verilmektedir.

Tablo 6.1: Rastrigin İçin Üretilen Rastgele Başlangıç Değerleri

x	y	x	y	x	y
0,414422	3,459031	4,946481	3,774889	-4,52091	-2,16013
5,025061	1,82763	-3,88392	-3,70267	-2,20695	1,444159
2,575229	2,250852	4,185291	-4,15246	-2,36549	4,405628
-3,07967	-3,32389	1,239004	-1,96155	-1,15323	-4,19119
-0,82314	-1,67503	3,610822	-3,22837	-2,32972	-4,91543
0,872693	-3,226	-2,23367	0,351977	-3,85543	-2,52805
3,415469	4,422788	0,419666	-1,63698	3,736424	1,302909
-2,53843	-0,22805	1,894312	2,555619	3,310572	0,613304
1,058851	-3,66965	2,750299	2,468887	-4,46854	-3,06095
-2,76263	-0,85854	-4,16062	0,486652	4,361015	2,087583
1,832154	-1,99289	3,026772	-2,7656	-0,76147	-0,49614
4,432218	3,462926	-1,50362	1,745546	-1,16034	-2,32348
-4,46576	0,191333	-2,25468	-2,48896	0,388895	-4,53985
-2,72854	-4,41434	5,099247	4,84939	1,713755	-0,52427
0,119462	1,434441	-1,33986	1,66405	0,293461	0,8076
-0,55698	1,190547	-1,35833	-0,51418	-4,49731	-3,974
-3,23252	0,961818	-4,38828	-4,44747		

Tablo 6.2'de Rastrigin fonksiyonundan alınan sonuçları ve sonuçları bulma süreleri bulunmaktadır. Sol taraftaki çözümler önerilen algoritmayı sağ taraf ise karşılaştırma algoritmasını göstermektedir. Tabloda bulunan **E** ifadesi 10'un katı anlamına gelmektedir. Bu durumda **E-11** ifadesi 10^{-11} değerini, **E+11** ifadesi 10^{+11} değerini ifade eder.

Tablo 6.2: Rastrigin Fonksiyonu Çözümleri ve Çözüm Süreleri

Önerilen Algoritma				Karşılaştırma Algoritması			
min_x	min_y	min_z	süre(sn)	min_x	min_y	min_z	süre(sn)
1,82E-08	3,72E-08	**0**	2,043613	-2,14E-06	-2,46E-06	**1,13E-11**	2,106013
4,76E-08	-3,18E-08	**3,55E-15**	1,981213	8,58E-07	3,15E-06	**1,13E-11**	2,012413
5,02E-09	-4,33E-08	**0**	2,043613	3,36E-07	1,74E-06	**3,32E-12**	1,981213
3,02E-08	2,06E-08	**0**	1,996813	-4,30E-07	2,06E-06	**4,70E-12**	1,981213
-2,00E-08	-1,83E-09	**0**	1,981213	3,01E-06	2,48E-06	**1,61E-11**	1,996813
2,90E-08	-2,87E-08	**0**	1,981213	1,80E-06	2,14E-06	**8,29E-12**	1,981213
-2,35E-08	1,66E-08	**0**	1,981213	-1,83E-06	1,85E-06	**7,21E-12**	1,996813
-2,92E-08	3,60E-08	**0**	1,981213	-2,30E-06	-2,10E-06	**1,03E-11**	1,996813
-5,81E-10	-2,94E-09	**0**	1,996813	-3,71E-06	6,67E-07	**1,50E-11**	2,012413
4,25E-08	-2,57E-08	**0**	2,012413	-1,67E-06	-2,98E-06	**1,23E-11**	2,074813
2,94E-08	-5,15E-08	**0**	2,043613	-2,25E-06	2,40E-06	**1,15E-11**	2,059213
-1,79E-08	1,52E-08	**0**	2,028013	1,36E-06	1,24E-06	**3,59E-12**	2,074813
1,69E-06	2,53E-06	**9,84E-12**	2,059213	-2,79E-05	-8,10E-05	**7,78E-09**	2,043613

-3,20E-08	1,87E-09	**0**	2,059213	-2,31E-06	3,22E-07	**5,78E-12**	2,074813
-1,32E-08	2,72E-08	**0**	2,059213	1,01E-08	9,37E-07	**9,31E-13**	2,012413
-8,94E-09	3,85E-08	**0**	2,012413	-4,04E-06	3,46E-06	**3,00E-11**	1,950013
1,67E-08	7,77E-09	**0**	2,043613	1,68E-06	1,71E-06	**6,10E-12**	2,074813
-2,40E-08	-2,36E-08	**0**	2,043613	4,68E-07	-2,04E-07	**2,77E-13**	2,059213
-3,78E-09	-4,41E-10	**0**	2,043613	1,32E-06	-1,16E-06	**3,27E-12**	2,043613
1,91E-08	3,89E-08	**0**	2,059213	-7,76E-07	-3,24E-06	**1,18E-11**	2,028013
2,53E-08	4,05E-08	**0**	2,137214	6,29E-07	4,84E-06	**2,53E-11**	2,059213
3,50E-08	2,01E-08	**0**	2,043613	3,68E-06	-3,02E-06	**2,41E-11**	2,059213
-6,32E-09	4,30E-08	**0**	2,199614	3,11E-06	-1,46E-06	**1,25E-11**	2,106013
-3,94E-08	9,40E-09	**0**	2,043613	4,89E-07	1,93E-07	**2,91E-13**	2,059213
-2,56E-08	3,47E-08	**0**	2,262014	-2,04E-06	-7,91E-07	**5,07E-12**	2,090413
-1,98E-09	-3,94E-08	**0**	2,184014	1,93E-06	-1,83E-06	**7,49E-12**	2,308815
8,12E-09	1,41E-08	**0**	2,558416	-5,09E-06	-3,02E-06	**3,72E-11**	2,620817
-2,24E-08	5,08E-08	**0**	2,480416	1,49E-06	-6,08E-07	**2,74E-12**	2,262014
-2,83E-09	3,43E-08	**0**	2,121614	1,94E-06	5,95E-06	**4,16E-11**	2,137214
-6,89E-09	-4,11E-10	**0**	2,028013	-2,53E-06	-2,28E-06	**1,23E-11**	2,012413
-2,94E-08	1,51E-09	**0**	2,059213	-6,95E-06	2,07E-06	**5,57E-11**	2,043613
-9,52E-09	-3,51E-08	**0**	2,074813	3,67E-07	1,54E-06	**2,66E-12**	2,121614
1,59E-08	-8,15E-09	**0**	2,074813	2,79E-06	-1,03E-06	**9,36E-12**	2,121614
-1,95E-08	3,60E-08	**0**	2,152814	-8,81E-07	1,48E-05	**2,32E-10**	2,168414
-3,10E-08	3,00E-08	**3,55E-15**	2,137214	3,12E-07	-1,90E-06	**3,94E-12**	2,386815
-4,43E-08	2,19E-08	**0**	2,090413	3,22E-05	8,71E-05	**9,14E-09**	2,184014
3,36E-08	2,35E-08	**0**	2,168414	-1,99E-06	-1,97E-06	**8,33E-12**	2,246414
1,49E-08	-5,78E-09	**0**	2,168414	1,10E-06	-9,19E-07	**2,18E-12**	2,121614
1,38E-09	-4,95E-08	**0**	2,168414	1,58E-06	1,66E-06	**5,54E-12**	2,012413
-1,97E-09	1,53E-08	**0**	1,996813	2,62E-06	-2,07E-06	**1,18E-11**	2,012413
-2,91E-08	1,68E-09	**0**	1,996813	-1,36E-07	8,26E-07	**7,43E-13**	1,996813
-6,83E-09	-2,68E-08	**0**	1,981213	-1,06E-05	-3,38E-05	**1,33E-09**	2,028013
2,46E-08	1,57E-08	**0**	2,090413	1,03E-05	2,70E-06	**1,20E-10**	2,293215
-1,68E-08	-1,18E-08	**0**	2,355615	-3,25E-06	3,07E-06	**2,12E-11**	2,199614
-4,60E-08	3,07E-08	**3,55E-15**	2,106014	1,06E-06	-1,08E-06	**2,44E-12**	2,137214
-2,40E-08	-3,75E-08	**0**	1,996813	1,11E-06	-3,61E-06	**1,51E-11**	1,996813
-1,04E-08	3,40E-09	**0**	2,106014	1,32E-06	-1,21E-06	**3,40E-12**	2,012413
-1,74E-08	1,35E-09	**0**	2,043613	1,09E-06	3,19E-08	**1,25E-12**	2,090413
-1,96E-07	-2,62E-07	**1,14E-13**	2,121614	-3,13E-06	-6,56E-08	**1,04E-11**	2,028013
3,15E-10	-5,06E-08	**0**	2,090413	8,27E-07	1,07E-06	**1,93E-12**	2,043613

Elde ettiğimiz sonuçlara göre saniyenin binde biri gibi bir farkla önerilen algoritma daha hızlı çalışmaktadır. Karşılaştırma algoritmasının ortalama çözüm süresi 2.09 sn olurken önerilen algoritmanın ortalama çözüm süresi 2.08 sn olmaktadır. Önerilen algoritmanın bulduğu sonuçlar karşılaştırma algoritmasının bulduklarına göre daha hassastır.

Tablo 6.3'te ise Rosenbrock fonksiyonunun çözümü için üretilen rastgele değerleri gösterilmektedir.

Tablo 6.3: Rosenbrock İçin Üretilen Rastgele Başlangıç Değerleri

x	y	x	y	x	y
-0,00179	0,085297	-0,60733	-0,04904	-0,43155	0,593216
0,59581	-0,99015	-1,14735	0,599034	-1,61385	-1,13585
1,417894	0,026898	-0,70345	-1,03533	-0,54721	-0,14763
-0,16514	-0,12944	0,058591	-0,81843	-1,27852	0,725169
1,854763	2,042388	0,937408	1,564599	-1,96313	0,717305
-1,98948	-1,44306	0,435206	-0,68226	0,989002	0,242149
-1,66467	-0,83965	0,088862	-0,08563	-1,23835	0,67414
-0,53662	-0,08858	1,959471	-1,68983	1,84865	-0,28472
-0,17289	-0,06279	-1,92387	1,914235	0,466336	0,529937
-0,99483	1,869926	-1,4455	1,21504	-0,43868	1,654327
1,063118	0,469915	1,780458	0,04984	-0,20364	1,289845
0,430013	0,44172	1,74325	1,5212	2,047172	-0,52786
1,930589	-1,18366	-1,91945	-0,95283	-1,11476	-2,00817
0,794797	1,840222	0,647534	0,494838	-1,89881	1,96506
1,573514	1,225775	-0,66762	1,811903	0,534151	0,051882
1,71697	-1,54948	1,348782	-1,52509	1,779922	-0,38935
1,111165	-1,14657	-0,59766	-1,11633		

Tablo 6.4'te Tablo 6.3'teki başlangıç noktalarına bağlı olarak elde edilen sonuçlar bulunmaktadır.

Tablo 6.4: Rosenbrock Fonksiyonu Çözümleri ve Çözüm Süreleri

Önerilen Algoritma				Karşılaştırma Algoritması			
min_x	min_y	min_z	süre(sn)	min_x	min_y	min_z	süre(sn)
0,998577	0,997385	**7,26E-06**	1,49761	0,998096	0,99619	**3,63E-06**	1,54441
0,999384	0,998772	**3,80E-07**	1,388409	1,003715	1,007423	**1,38E-05**	1,48201
-21631,8	4,68E+08	**4,68E+08**	2,808018	0,999487	0,998953	**3,09E-07**	1,404009
0,998522	0,997215	**5,09E-06**	1,357209	0,998983	0,997687	**8,85E-06**	1,404009
0,999978	0,999965	**8,93E-09**	1,388409	0,999013	0,99794	**1,74E-06**	1,372809
-21631,8	4,68E+08	**4,68E+08**	2,808018	1,001426	1,002766	**2,79E-06**	1,435209
0,999873	0,999772	**8,63E-08**	2,730018	0,999704	0,999329	**7,04E-07**	1,372809
0,998338	0,996557	**4,25E-06**	1,388409	0,998388	0,996762	**2,62E-06**	1,372809
1,000169	1,000309	**1,12E-07**	1,388409	1,000707	1,001293	**1,96E-06**	1,372809
0,99972	0,999462	**1,24E-07**	1,404009	0,99961	0,999174	**3,71E-07**	1,388409
0,999742	0,999521	**2,11E-07**	1,388409	0,997308	0,994693	**7,73E-06**	1,357209
1,000115	1,000328	**9,80E-07**	2,714417	0,998574	0,997033	**3,41E-06**	1,326009
0,999608	0,998981	**5,63E-06**	2,745618	1,001121	1,002133	**2,44E-06**	1,372809
1,001065	1,002167	**1,26E-06**	1,341609	0,99995	0,999859	**1,73E-07**	1,357209

0,997213	0,994803	**2,13E-05**	1,388409	0,998439	0,996717	**5,11E-06**	1,357209
1,003144	1,0064	**1,09E-05**	2,714417	0,999062	0,998161	**1,01E-06**	1,341609
1,000065	1,000418	**8,30E-06**	2,683217	0,99923	0,998267	**4,29E-06**	1,341609
1,001124	1,002354	**2,37E-06**	1,372809	1,002253	1,004289	**1,00E-05**	1,51321
1,000087	1,000147	**8,02E-08**	1,357209	1,000467	1,000771	**2,85E-06**	1,404009
0,999457	0,999057	**2,36E-06**	2,745618	0,997744	0,995284	**9,44E-06**	1,357209
0,999472	0,999021	**8,63E-07**	1,372809	0,999632	0,999221	**3,15E-07**	1,372809
1,000829	1,001648	**7,01E-07**	1,357209	1,004789	1,009534	**2,34E-05**	1,357209
0,999584	0,999041	**1,79E-06**	1,341609	0,999849	0,999671	**9,90E-08**	1,326009
0,999755	0,999495	**8,07E-08**	1,326008	0,999863	0,999696	**1,06E-07**	1,341609
1,000594	1,001215	**4,23E-07**	2,667617	0,993881	0,987573	**4,26E-05**	1,326009
0,999658	0,999451	**1,93E-06**	2,667617	0,999355	0,99853	**3,68E-06**	1,341609
0,99892	0,998055	**5,68E-06**	1,341609	0,998537	0,99687	**6,42E-06**	1,341609
0,999127	0,998194	**1,13E-06**	2,683217	0,999733	0,999371	**9,54E-07**	1,357209
0,999319	0,998594	**6,52E-07**	1,341609	1,000916	1,001693	**2,81E-06**	1,341609
0,999788	0,999592	**7,47E-08**	2,683217	0,99635	0,99286	**1,55E-05**	1,404009
0,999411	0,998786	**4,81E-07**	1,404009	0,997907	0,995881	**4,79E-06**	1,357209
0,999638	0,999222	**4,28E-07**	1,372809	1,002922	1,00564	**1,30E-05**	1,357209
0,998661	0,997294	**1,89E-06**	2,730018	0,998693	0,99731	**2,30E-06**	1,357209
1,000821	1,002157	**2,71E-05**	1,341609	0,996998	0,993562	**2,86E-05**	1,341609
0,999656	0,998929	**1,48E-05**	1,357209	1,000267	1,000487	**2,85E-07**	1,341609
0,99969	0,999546	**2,84E-06**	2,652017	0,999887	0,999699	**5,79E-07**	1,326008
1,001422	1,003271	**2,00E-05**	1,326008	0,999546	0,999028	**6,06E-07**	1,326009
1,000533	1,00044	**3,94E-05**	1,341609	0,998733	0,997324	**3,64E-06**	1,326008
1,000765	1,001432	**1,57E-06**	2,636417	0,997469	0,995015	**6,90E-06**	1,326008
0,999826	0,99967	**6,51E-08**	2,730018	0,998788	0,997445	**3,20E-06**	1,388409
1,001594	1,003025	**5,29E-06**	1,357209	0,999786	0,999562	**5,41E-08**	1,341609
0,999643	0,999286	**1,27E-07**	2,730017	1,000875	1,001608	**2,81E-06**	1,357209
1,000874	1,001568	**4,07E-06**	1,357209	1,000609	1,001209	**3,77E-07**	1,357209
0,999164	0,99826	**1,16E-06**	1,435209	0,999384	0,998747	**4,30E-07**	1,341609
0,99958	0,999243	**8,71E-07**	1,388409	0,998076	0,995965	**7,36E-06**	1,326008
0,998786	0,997748	**4,48E-06**	2,667617	0,999537	0,999029	**4,28E-07**	1,357209
1,000176	1,000346	**3,55E-08**	2,745618	0,999149	0,998208	**1,54E-06**	1,341609
1,000944	1,001731	**3,38E-06**	1,372809	-1,6194	2,62236	**6,861252**	1,357209
1,001194	1,002374	**1,45E-06**	1,357209	0,999784	0,999502	**4,79E-07**	1,357209
0,999357	0,998895	**3,66E-06**	2,698817	0,997397	0,994538	**1,37E-05**	1,357209

Rosenbrock fonksiyonunun çözümünden elde ettiğimiz sonuçlara göre ise karşılaştırma algoritması çok az bir farkla daha hızlı olmaktadır. Karşılaştırma algoritmasının ortalama çözüm süresi 1.37 sn olurken önerilen algoritmanın çözüm süresi 1.9 sn olmaktadır. Elde edilen sonuçlar incelendiğinde ise sonuçların hemen hemen aynı olmasına rağmen önerilen algoritma daha hassas sonuçlar elde etmiştir.

Tablo 6.5 ise Griewangk fonksiyonunun çözümü için üretilen rastgele değerleri göstermektedir.

Tablo 6.5: Griewangk Fonksiyonu Çözümleri ve Çözüm Süreleri

x	y	x	y	x	y
449,2999	-465,85	133,4814	135,3945	153,6292	1,166554
-273,202	-262,665	345,1364	465,2657	175,7778	-20,4157
-161,556	244,1957	-560,065	-299,098	-39,4779	-520,74
487,4203	-254,169	323,9972	-194,642	84,53378	-238,934
-565,14	-523,865	-227,233	-421,423	88,39209	-454,628
-408,908	-236,474	-369,114	423,9601	-476,886	181,1761
-442,783	-275,128	-582,325	-201,258	-467,524	285,4149
-431,493	570,2428	-171,314	-412,015	266,6015	-562,617
-539,855	326,0999	-40,0249	356,8216	-214,422	438,758
-567,339	-364,259	-192,878	-469,793	557,0893	471,879
104,0779	401,6937	556,0343	-423,938	-42,4605	-226,624
25,95572	344,1872	-541,839	195,9196	15,38425	-207,258
-570,18	226,5861	-33,4538	-36,7465	240,9133	248,3451
-484,669	-234,193	487,2795	386,47	-576,991	230,4726
-486,559	337,1844	190,4178	-214,363	61,80231	317,1223
-545,726	581,9787	-498,97	94,3159	-71,3362	-182,867
196,0687	-85,0692	-279,462	301,3264		

Tablo 6.6'da Tablo 6.5'teki başlangıç noktalarına bağlı olarak elde edilen sonuçlar bulunmaktadır.

Tablo 6.6: Griewangk Fonksiyonu Çözümleri ve Çözüm Süreleri

Önerilen Algoritma				Karşılaştırma Algoritması			
min_x	min_y	min_z	süre(sn)	min_x	min_y	min_z	süre(sn)
-0,73663	-1,56795	0,00102	2,152814	0,660088	-0,99927	0,000501	2,074813
0,019827	0,653612	0,000139	2,121614	-0,15687	1,111157	0,000413	2,074813
0,053626	-1,01807	0,000339	2,106013	1,910213	2,507635	0,003519	2,074813
0,049614	0,709857	0,000165	2,106014	-0,97064	-1,38069	0,001001	2,074813
0,003173	-0,00524	1,30E-08	2,121614	0,030076	0,527276	9,10E-05	2,059213
-0,00859	0,05986	1,20E-06	2,106014	0,428412	0,087749	7,63E-05	2,090413
1,133974	-0,01691	0,000517	2,184014	-0,11256	1,001839	0,000332	2,464816
0,003173	-0,00524	1,30E-08	2,308815	-0,45285	-0,16014	9,09E-05	2,184014
-0,02556	0,411365	5,55E-05	2,246414	-0,07695	-0,82829	0,000226	2,106013
0,003173	-0,00524	1,30E-08	2,558416	0,893924	-1,50709	0,001062	2,215214
-4,79264	-4,41849	0,015601	2,168414	-0,09496	-1,18955	0,000465	2,152814
1,914506	2,132591	0,002958	2,355615	-1,0765	-0,59132	0,00058	2,184014
0,688786	1,421151	0,00085	2,215214	-1,64508	-3,51711	0,005123	2,168414
-2,11925	-2,08955	0,003231	2,277615	-1,32639	-1,23308	0,001204	2,246414

1,135893	-2,25571	0,002179	2,293215	0,635479	2,498883	0,002199	2,152814
0,003173	-0,00524	1,30E-08	2,230814	0,23518	1,1356	0,000443	2,137214
-0,70831	0,064461	0,000203	2,168414	-0,03287	-0,44682	6,55E-05	2,184014
1,78706	2,002576	0,002593	2,168414	-0,76382	-0,79409	0,00044	2,152814
2,092058	-2,4025	0,003643	2,152814	0,545872	-0,23123	0,000137	2,074813
4,09071	4,355603	0,012915	2,137214	-0,26842	-0,05741	3,01E-05	2,059213
0,69377	-1,51146	0,000939	2,106013	0,026263	0,729092	0,000174	2,059213
0,565886	1,079127	0,000509	2,106013	0,017285	0,310215	3,15E-05	2,074813
0,616766	1,783096	0,00119	2,106014	0,11883	-1,58805	0,000828	2,059213
0,01158	0,01082	9,21E-08	2,106014	-0,35555	-3,8742	0,004946	2,059213
-0,04218	-1,38196	0,000624	2,090413	0,046345	-0,56573	0,000105	2,106014
1,501624	-2,08425	0,002324	2,152814	0,361705	1,627714	0,000917	2,121614
-3,55948	-2,63746	0,007364	2,152814	1,049353	3,404283	0,004223	2,121614
0,003173	-0,00524	1,30E-08	2,137214	-0,84039	-2,25195	0,001938	2,106014
-0,28672	2,554756	0,002162	2,137214	-0,3086	-1,35161	0,000634	2,074813
0,000318	-9,53E-06	4,07E-11	2,106014	-0,44433	-2,36457	0,001903	2,074813
-4,52181	-5,02798	0,016463	2,168414	0,171286	-1,98522	0,001297	2,121614
-0,04365	-0,45503	6,83E-05	2,137214	0,666881	-0,19356	0,000191	2,106014
1,16889	-3,29292	0,004086	2,121614	-0,03301	-0,48687	7,78E-05	2,152814
0,013783	-0,26893	2,37E-05	2,137214	-0,9265	-3,62526	0,004632	2,137214
-0,17694	-0,83322	0,000239	2,215214	-2,69681	-5,63215	0,013268	2,137214
-0,66568	-2,76219	0,002667	2,168414	-2,00879	-0,60817	0,001744	2,121614
0,003743	-0,2103	1,44E-05	2,152814	-0,90405	2,030905	0,001674	2,074813
0,612748	-0,53891	0,000246	2,106014	-0,83028	-3,18621	0,003588	2,090413
0,050518	-0,08949	3,64E-06	2,215214	-0,64548	-1,96735	0,00143	2,090413
-0,01007	-0,2849	2,65E-05	2,152814	0,073727	0,855332	0,000241	2,059213
-0,87424	0,957788	0,000607	2,168414	0,021076	-0,53789	9,45E-05	2,090413
0,003173	-0,00524	1,30E-08	2,137214	-0,09182	-1,35237	0,0006	2,106014
0,081991	1,313731	0,000566	2,168414	-0,71814	1,200092	0,000677	2,106013
0,003173	-0,00524	1,30E-08	2,152814	0,15086	-1,07466	0,000386	2,090413
-0,23425	-1,29571	0,00057	2,168414	0,08589	-1,24148	0,000506	2,137214
-0,29087	0,630026	0,000163	2,121614	0,57668	2,158156	0,001653	2,106014
-0,00322	-0,02439	1,98E-07	2,137214	0,431864	-1,35902	0,000677	2,090413
-2,35504	-0,02308	0,002231	2,121614	2,153646	0,660307	0,002008	2,090413
0,522032	0,454551	0,000177	2,090413	0,327741	0,06803	4,47E-05	2,090413
0,392902	-2,28113	0,001759	2,090413	-0,13172	-0,92741	0,000287	2,074813

Griewangk fonksiyonunun çözümünden elde ettiğimiz sonuçlara göre iki algoritmanın çözüm süreleri hemen hemen aynı çıkmıştır. Karşılaştırma algoritmasının ortalama çözüm süresi 2.12 sn olurken önerilen algoritmanın çözüm süresi 2.17 sn olmaktadır. Elde edilen sonuçlara bakıldığında ise önerilen algoritmanın sonuçları daha hassas çıkmıştır.

7. TARTIŞMALAR VE SONUÇ

Bu çalışmada numerik bir yöntemle sezgisel bir yöntemin birleştirilerek kesin çözüm bulan hızlı bir algoritma elde edilebileceği tezi ortaya atılmıştır.

BFGS yöntemi türev tabanlı bir yöntemdir. Türevin sıfıra eşit olduğu noktaya kadar ilerler ve sıfır olduğu noktada kalır. Bu sebeple çözüme başladığı ilk noktaya en yakın ekstremum noktaya takılır. Bulduğu sonuç ise belirlenen hassasiyete bağlı olmakla birlikte kesin çözümü garanti edebilir. Çalışmamızda BFGS için hassasiyet 10^{-3} belirlenmiştir. Yeni bulunan çözümler arasındaki değişim, bu çözümlerin fonksiyon yarattığı değişim ve sonuca yakınsama hassasiyeti olarak belirtilen değer kullanılmıştır.

PSO algoritması az parametre gerektiren ve dolayısıyla daha hızlı sonuca ulaşan bir sezgisel yöntemdir. Çözüme belirlenen parçacık sayısına bağlı olarak bir çok noktadan yola çıkarak başlar. Parçacıklar arasında ekstremum noktaya en yakın olanı çözüm kabul eder. PSO algoritmaları genellikle parçacıklara 1000, 2000 veya 3000 iterasyon yaptırır. İterasyon sayısındaki değişiklikler problem tipine göre kazanç da olabilir kayıp da olabilir. Çalışmamızda 2000 iterasyon seçilmiş olup iterasyon sayısı arttıkça performans düşüşü yaşanmıştır.

Çalışmamızda BFGS ve PSO yöntemleri birleştirilmiş. BFGS ile bir yerel minimuma ulaşılmış ve fonksiyonda bu çözüme eşit diğer noktalar PSO ile tespit edilmiştir. Ardından bulunan diğer noktaların herhangi birinden başka bir yerel minimuma ulaşılmıştır. Bu döngü durdurma kriterleri sağlanana kadar devam etmektedir.

Yapılan deneyler ortaya atılan tezin sonuç bulma yeteneğini ispatlamıştır. Karşılaştırma amaçlı kullanılan algoritmanın bulduğu sonuçlara göre daha hassas ve kesin sonuçlar elde etmiştir.

Performans karşılaştırmalarından elde edilen verilere göre karşılaştırma algoritması milisaniye düzeyinde daha hızlı olmaktadır. Hız farkları fonksiyonun karmaşıklığına bağlı olarak değişmektedir. Örneğin; Rastrigin gibi daha basit yapılı

test fonksiyonlarında milisaniye düzeyinde önerilen algoritma daha hızlıyken Griewangk gibi nispeten daha zorlu fonksiyonlarda karşılaştırma algoritmasının milisaniye düzeyinde daha hızlı olduğu tespit edilmiştir.

Elde edilen verileri özetlersek; önerilen algoritma performans açısından muadillerine eşdeğer sonuçlar vermiş olup çözüme ulaşma konusunda daha başarılı olduğunu ispatlamıştır. Öte yandan amaç fonksiyonu karmaşıklaştıkça önerilen algoritmanın yavaşladığı da gözlemlenmiştir. BFGS gibi türev bilgisi gerektiren bir yöntem kullanıldığı için önerilen algoritmanın çözeceği problemlerin türevlenebilir bir amaç fonksiyonuna sahip olması gerekmektedir. Bundan dolayı Gezgin Satıcı Problemi ve Araç Rotalama Problemi gibi amaç fonksiyonu olmayan problem çözümlerine uygulanamaz.

Sonuç olarak BFGS gibi numerik yöntemlerle PSO gibi sezgisel yöntemlerin bir arada kullanılması çözüm kesinliği için uygun çözüm olabilir. Belli uygulamalarda ise performans olarak bir çözüm olabilir. Bu çalışmada amaçlandığı gibi başka numerik yöntemlerle sezgisel yöntemler melezlenebilir. Ayrıca yapılan literatür araştırması ve önerilen algoritma melez algoritmaların standart sezgisellere ve numerik yöntemlere göre daha iyi sonuçlar elde ettiğini ispatlamıştır.

8. KAYNAKLAR

[1] **Karaboğa , D.,** (2011) Yapay Zeka Optimizasyon Algoritmaları, 2. baskı, Nobel Yayın Dağıtım, Ankara, 231s.

[2] **Davidon, W., C.,** (1991) Variable Metric Method for Minimization. SIAM J. Optimization, Vol. 1, No. 1, S.1-17.

[3] **İplikçi, S.,** (2010) Optimizasyon Teknikleri Ders Notları.

[4] **Molga, M., Smutnicki, C.,** (2005) Test Functions for Optimization Needs.

[5] **Keskintürk, T.,** (2006/1) Diferansiyel Gelişim Algoritması. İstanbul Ticaret Üniversitesi Fen Bilimleri Dergisi, Yıl: 5, Sayı: 9, s.85-99.

[6] **Cura, T.,** (2008) Doğrusal Olmayan Küresel Optimizasyon Problemleri için Tabu Arama Algoritmasının Kullanılması, İstanbul Üniversitesi İşletme Fakültesi Dergisi, Cilt: 37, Sayı: 1, s.22-38.

[7] **Bell, J., E., McMullen P., R., (**2004) Ant Colony Optimization Techniques for the Vehicle Rounting Problem, Advanced Engineering Informatics, 18 s.41-48.

[8] **Kılıç, S., Kahraman, C.,** (2009) Bulanık Karar Ortamında Karınca Kolonisi Optimizasyonu Yöntemiyle Araç Rotalama, İstanbul Teknik Üniversitesi Mühendislik Dergisi, Cilt: 8, Sayı: 4, s.160-172.

[9] **Ateş, E., (**2012) Karınca Kolonisi Optimizasyonu Algoritmaları ile Gezgin Satıcı Probleminin Çözümü ve 3 Boyutlu Benzetimi, Lisans Tezi, Ege Üniversitesi Mühendsilik Fakültesi Bilgisayar Mühendisliği Bölümü, İzmir, 41s.

[10] **Szeto, W., Y., Wu, Y., Ho, C., S.,** (2011) An Artificial Bee Colony Algorithm for the Capacitated Vehicle Routing Problem, European Journal of Operational Research, 215 s.126-135.

[11] **Kennedy, J., Eberhart, R.,** (1995) Particle Swarm Optimization, IEEE.

[12] **Khare, A., Rangnekar, S.,** (2013) Particle Swarm Optimization: A Review, Applied Soft Computing, Volume 13, Issue 5, s.2997-3006.

[13] **Liu, Y., Zheng, Q., Shi, Z., Lu, J.,** (2007) Center Particle Swarm Optimization, Neurocomputing, 70 s.672-679.

[14] **Shi, Y., Eberhart, R.,** (1998) A Modified Particle Swarm Optimizer, IEEE.

[15] **Perez, R., E., Behdinan, K.,** (2007) Particle Swarm Approach for Structural Design Optimization, Computers and Structures, 85 s.1579-1588.

[16] **Shi, Y., Liu, H., Gao, L., Zhang, G.,** (2011) Cellular Particle Swarm Optimization, Information Sciences, 181 s.4460-4493.

[17] **Coelho, S., L.,** (2010) Gaussian Quantum-Behaved Particle Swarm Optimization Approaches for Constrained Engineering Design Problems, Expert Systems with Applications, 37 s.1676-1683.

[18] **Mousa, A., A., El-Shorbagy, M., A., Abd-El-Wahed, W., F.,** (2011) Local Search Based Hybrid Particle Swarm Optimization Algorithm for Multiobjective Optimization, 3 s.1-14.

[19] **Nawi, N., M., Ransing, M., R., Ransing, R., S.,** (2006) An Improved Learning Algorithm Based on BFGS Method For Back Propagation Neural Nteworks, IEEE.

[20] **Xia, Y., Chan, K., W., Liu, M.,** (2005) Improved BFGS Method for Optimal Power Flow Calculation with Transient Stability Constraints, IEEE.

[21] **Semeter, J., Mendillo, M.,** (1997) A Nonlinear Optimization Technique for Ground-Based Atmospheric Emission Tomography, IEEE.

9. ÖZGEÇMİŞ

Ad Soyad	: Mehmet CENGİZ
Doğum Yeri ve Tarihi	: Ankara – 08.02.1989
Lisans Üniversite	: Pamukkale Üniversitesi
Elektronik posta	: mcengiz07@pau.edu.tr
İletişim Adresi	: Atatürk mah. Kazımkarabekir cad. No: 93-3 UŞAK
Çalışmalar	:

Mehmet CENGİZ, "Doğal Dilde İşleme", Lisans Tezi (2011).