BİLECİK ÜNİVERSİTESİ

Fen Bilimleri Enstitüsü

Bilgisayar Mühendisliği Anabilim Dalı

GELİŞTİRİLMİŞ YERÇEKİMSEL ARAMA ALGORİTMASI

Nihan Kazak
Yüksek Lisans Tezi

Tez Danışmanı
Yrd. Doç. Dr. Alpaslan DUYSAK

BİLECİK, 2011

 **BİLECİK ÜNİVERSİTESİ** **FEN BİLİMLERİ ENSTİTÜSÜ**	**YÜKSEK LİSANS** **JÜRİ ONAY FORMU**

Bilecik Üniversitesi Fen Bilimleri Enstitüsü Yönetim Kurulu'nun ……..……tarih ve …… sayılı kararıyla oluşturulan jüri tarafından …… tarihinde tez savunma sınavı yapılan Nihan Kazak'ın "Geliştirilmiş Yerçekimsel Arama Algoritması" başlıklı tez çalışması Bilgisayar Mühendisliği Anabilim Dalında YÜKSEK LİSANS tezi olarak oy birliği/oy çokluğu ile kabul edilmiştir.

JÜRİ

ÜYE

(TEZ DANIŞMANI) : Yrd. Doç. Dr. Alpaslan DUYSAK

ÜYE : Yrd. Doç. Dr. Cihan KARAKUZU

ÜYE : Yrd. Doç. Dr. Pakize ERDOĞMUŞ

ONAY

Bilecik Üniversitesi Fen Bilimleri Enstitüsü Yönetim Kurulu'nun ……/……/……tarih ve ……/…… sayılı kararı.

İMZA/MÜHÜR

ÖZET

Optimizasyon teknikleri birçok mühendislik probleminin çözümünde kullanılmaktadır. Son zamanlarda klasik optimizasyon yöntemlerine ek olarak sezgisel optimizasyon yöntemleri, verilen şartlar altında en iyi, optimum, sonucu bulmak için geliştirilmiştir. Yerçekimsel arama algoritması (YAA) sezgisel optimizasyon yöntemlerinden biridir.

Yapılan bu tez kapsamında, YAA temel alınarak daha etkin, daha verimli ve daha hızlı optimizasyon algoritmalarının geliştirilmesi amaçlanmıştır. Bu amaçla, nesnelerin deformasyonunda sıkça kullanılan kütle-yay sistemleri metodu incelenmiş ve YAA içine ekilmiştir. Ek olarak YAA metodu daha fazla yüzey taraması yapacak şekilde modifiye edilmiştir. Geliştirilen algoritmalar test edilerek, sonuçları YAA ile karşılaştırılmıştır. Yapılan testler sonucunda algoritmaların daha etkin ve daha verimli sonuçlar verdiği fakat daha yavaş çalıştığı gözlemlenmiştir.

Anahtar Kelimeler

Optimizasyon, GSA, MSS-GSA, Üye-Uydu Algoritması

ABSTRACT

Optimization techniques are used to solve many engineering problems. Recently, in addition to the classical optimization methods, heuristic optimization methods have been developed to find the best, the optimum, result under the given conditions. Gravitational search algorithm (GSA) is one of the heuristic optimization methods.

Within the scope of this thesis, it is aimed to develope GSA based optimization algorithms which are more effective, more efficient and faster. For this purpose, mass spring system (MSS) method that is widely used in deformation of the objects is investigated and integrated to GSA. In addition, GSA method is modified to make more surface scan. Developed algorithms are tested and the results are compared with GSA. As a results of the tests, it has been observed that the algorithms give more effective and more efficient results but run more slowly.

Keywords

Optimization, GSA, MSS-GSA, Member-Satellite Algorithm

TEŞEKKÜR

Tez çalışmam boyunca bilgisini ve desteğini benden hiçbir zaman esirgemeyen değerli danışmanım Yrd. Doç. Dr. Alpaslan DUYSAK'a, çalışmalarımda büyük yardımları bulunan Yrd. Doç. Dr. Hasan TEMURTAŞ'a, bana olan inancını ve desteğini manevi olarak her zaman hissettiğim Yrd. Doç. Dr. Resul KARA ve Yrd. Doç. Dr. Pakize ERDOĞMUŞ'a, Bilecik Üniversitesi Bilgisayar Mühendisliği bölüm hocalarıma, çalışma arkadaşlarıma ve maddi manevi her türlü desteklerini benden hiçbir zaman esirgemeyen aileme teşekkürlerimi sunarım.

İÇİNDEKİLER

Sayfa No

ÇİZELGELER DİZİNİ

Sayfa No

ŞEKİLLER DİZİNİ

Sayfa No

KISALTMALAR DİZİNİ

DGA	: Diferansiyel Gelişim Algoritması
GA	: Genetik Algoritma
YAA	: Yerçekimsel Arama Algoritması (Gravitational Search Algorithm)
KKA	: Karınca Kolonisi Algoritması
MSS	: Kütle Yay Sistemi (Mass Spring System)
MSS-GSA	: Kütle Yay Sistemi – Yerçekimsel Arama Algoritması
PSO	: Parçacık Sürü Optimizasyonu
SEM	: Sonlu Elemanlar Metodu

1. GİRİŞ

Optimum kelimesi en uygun, en elverişli manasına gelmektedir. Optimizasyonun kelime anlamı ise en uygun duruma getirmedir. Bir sistemin, verilen şartlar altında ihtimal dâhilindeki en iyi çözümünü elde etme işlemi olarak tanımlanabilir. Optimizasyon, sistemi mümkün olan en iyi duruma getirirken bazı matematiksel yöntemler kullanır.

Bir sistem genel olarak ele alındığında verilen girdilere göre bazı çıktılar elde edilmektedir. İstenilen sonuçları elde etmenin çok sayıda alternatif çözümü olabilir. Bu çözümler arasında en iyisinin en kısa sürede bulunması bir optimizasyon işlemidir. Tasarlanan sistemden beklentiler arttığında, sistem daha karmaşık bir hale gelir ve o problemin çözümü daha zorlaşır. Literatürde bu tür problemlerin çözümünde bazı matematiksel bağıntılar kullanılarak klasik optimizasyon yöntemleri kullanılmıştır. Klasik optimizasyon yöntemlerinin kullanılmasının dezavantajları:

- Probleme özel olması,
- Problemin matematiksel fonksiyonlarla tanımlanması gerekliliği,
- Zaman alması, performansının düşük olmasıdır.

Klasik optimizasyon yöntemlerinin yukarıda sıralanan dezavantajlarını gidermek amacıyla bilim adamları son yıllarda çalışmalarını giderek arttırmıştır. Çözüme yönelik optimizasyon yöntemlerinin geliştirilmesi tabiatta gerçekleşen olayların modellenmesine yönelik ortaya atılan yapay zeka kavramının ortaya çıkmasına neden olmuştur. Optimizasyon yöntemleri hakkında detaylı bilgi vermeden önce bazı temel kavramları açıklamak faydalı olacaktır.

1.1. Temel Kavramlar

Karar parametreleri, optimize edilecek olan fonksiyonun aldığı parametrelerdir. Karar parametrelerine bağlı tanımlanan fonksiyon amaç fonksiyonunu temsil eder. Amaç fonksiyonu minimize edilecek maliyet fonksiyonu veya maksimize edilecek kar fonksiyonu olarak tanımlanır. Karar parametrelerini kullanarak yapılan minimizasyonda elde edilen değer düşük, maksimizasyonda ise büyüktür. Parametrelerin değer alması üzerine konulan kısıtlar sınırlayıcılar olarak adlandırılır. Bir fonksiyona bir veya birden fazla sınırlama konulabilir.

1.2. Optimizasyon Problemlerinin Sınıflandırılması

Optimizasyon problemleri sınırlamalı, sınırlamasız, doğrusal, doğrusal olmayan, tamsayı, tamsayı olmayan, sürekli, ayrık ve quadratik optimizasyon problemleri olarak sınıflandırılır (Karaboğa, 2004). Optimizasyon problemlerinin karar değişkenlerine göre sınıflandırılması ile amaç ve sınırlama fonksiyonlarına göre sınıflandırılması Çizelge 1.1. ile Çizelge 1.2.'de gösterilmiştir.

Çizelge 1.1. Optimizasyon problemlerinin amaç ve sınırlama fonksiyonlarına göre sınıflandırılması.

<table>
<tr><th colspan="2">Doğrusal Optimizasyon</th><th>Doğrusal Olmayan Optimizasyon</th><th>Quadratik Optimizasyon</th></tr>
<tr><td colspan="2">Amaç ve sınırlama fonksiyonları lineerse</td><td rowspan="3">Amaç ve sınırlama fonksiyonlarından herhangi biri ya da ikisi de lineer değilse</td><td rowspan="3">Amaç fonksiyonu quadratik, sınırlama fonksiyonu lineerse</td></tr>
<tr><td>Tamsayı Programlama</td><td>Tamsayı Olmayan Programlama</td></tr>
<tr><td>Değişkenleri negatif olmayan tamsayı değerler alıyorsa</td><td>Değişkenleri negatif olmayan tamsayı değerler almıyorsa</td></tr>
</table>

Çizelge 1.2. Optimizasyon problemlerinin karar değişkenlerine göre sınıflandırılması.

Sınırlamalı Optimizasyon	Sınırlamasız Optimizasyon	Sürekli Optimizasyon	Ayrık Optimizasyon
Karar parametrelerinde belli bir sınır ya da sınırlamalar mevcutsa	Karar parametrelerinde belli bir sınır ya da sınırlamalar mevcut değilse	Tasarım değişkenlerinin alacağı değerler sürekli değerler ise	Ayrık niceliklerin optimal olarak düzenlenmesi, gruplanması veya seçilmesi problemi

1.3. Optimizasyon Yöntemlerinin Sınıflandırılması

Optimizasyon problemlerinin çözümü için geliştirilen yöntemler genel olarak iki grupta incelenebilir:

- Doğrudan metotlar(Araştırma metotları),
- Dolaylı metotlar.

Dolaylı metotlarda ilk olarak optimallik kriterleri belirlenir, daha sonra bölgesel optimuma aday noktalar için problem çözülmektedir. Araştırma metotlarında ise tahmini bir başlangıç çözümü ile araştırmaya başlanır ve algoritma tarafından başlangıç çözümü iteratif olarak geliştirilir. Diğer bir deyişle optimum çözümleri bulmak amacıyla çözüm uzayı araştırılır (Erol, 2004).

1.4. Sezgisel Algoritmalar

Herhangi bir problemde hedefe ulaşabilmek için alternatif çözüm yolları değerlendirilerek, içlerinden en etkili olana karar vermek amacı ile tasarlanan algoritmalara sezgisel algoritmalar adı verilir. Sezgisel algoritmalarla, arama uzayında yapılan araştırma sonucunda, en iyi çözüme yakınsama garanti edilir. Fakat kesin çözüm garanti edilemez.

Sezgisel algoritmalara gerek duyulmasının sebepleri:

- Kesin çözümü elde etme işleminin tanımlanamadığı optimizasyon problemlerinin varlığı,
- Karar vericiler için sezgisel algoritmalar diğer algoritmalara göre daha kolay anlaşılır olabilir.
- Kesin çözümü elde etme işlemi ve öğrenme amaçlı kullanılabilir.
- Matematiksel tanımlamalarla ifade edilen gerçek dünya problemlerinde amaçların belirlenmesi, sınırlamaların oluşturulması, test edilecek alternatiflerin seçimi ve problem verilerinin toplanma biçimi ihmal edilebilir. Herhangi bir ihmal sonucunda model parametrelerinin belirlenmesinde karşılaşılan hata, sezgisel algoritmalarla elde edilen alt optimal çözümden çok daha büyük olabilir.

Sezgisel algoritmaların değerlendirilmesi için kriterler:

- Çözüm kalitesi ve hesaplama zamanı arasındaki ilişki kontrol edilebilmelidir.
- Algoritma prensipleri basit olmalı ve genel olarak uygulanabilir olmalıdır.
- Algoritmalar, yapılan değişiklikleri kolayca karşılayabilmelidir.
- Her zaman yüksek kaliteli ve kabul edilebilir çözümleri üretebilme kabiliyetine sahip olmalıdır.
- Algoritma kolayca analiz edilebilmelidir.
- Algoritma içinde insan-makine etkileşimi olmalıdır. İyi bir kullanıcı arayüzü kullanılarak grafiksel olarak sergilenebilmelidir (Karaboğa, 2004).

Sezgisel yöntemler klasik ve metasezgisel olmak üzere ikiye ayrılabilir (Kaya ve Engin, 2009).

1.5. Metasezgisel Algoritmalar

Metasezgisel yöntemler çözüm uzayında etkili bir şekilde arama yapmak için, farklı yapılardaki alt kademe sezgisel algoritmaların zekice birleştirilmesi ile oluşturulmuş iteratif problem çözme yöntemlerdir. Bu yöntemler her iterasyonda bir çözümden veya çözüm koleksiyonundan yola çıkarak yeni çözümler üretirler. Çoğu metasezgisel yaklaşım, çözüm uzayında stokastik fakat bilinçli bir şekilde arama yapar (Blum ve Roli, 2003).

Metasezgisel yöntemler için yapılan bazı genellemeler (Erol, 2004):

- Metasezgisel yöntemler arama süreci sırasında kılavuzluk yapan stratejilerdir.
- Genel amaç, arama uzayını etkili bir şekilde araştırıp optimum veya optimuma yakın sonuçlar elde etmektir.
- Metasezgisel yöntemler yerel arama tekniklerinden, karmaşık öğrenme yöntemlerine kadar yaygınlık gösterir.
- Metasezgisel algoritmalar yaklaşık yöntemlerdir ve genellikle deterministik değildir.
- Arama uzayında yerel optimum noktalardan takılıp kalmayı engelleyecek mekanizmaları içinde barındırırlar.
- Metasezgisel teknikler probleme özel yöntemler değildirler. Genellikle tüm kombinatoryel problemlere uygulanabilirler.

- Günümüzde gelişmiş metasezgisel algoritmalarda arama sırasında kılavuzluk etmesi için hafıza tabanlı süreçler bulunmaktadır.

Optimizasyon problemlerinden olan ayrık problemler, optimal çözüme ulaşma yolunda yapılan hesaplamalar bakımından oldukça zor problemlerdir. Bu nedenle, araştırmacılar bu problemlerin çözümü için büyük bir çaba harcamışlardır. Çok boyutlu problemlerin çözümünde optimal sonuca uygun bir sürede yaklaşmayı başarabilen yaklaşık algoritmaları geliştirmek ve bu algoritmaları kullanmak araştırmacıların ilgi odağı haline gelmiştir. Bundan dolayı zeki ve etkili sezgisel algoritmalar gün geçtikçe önem kazanmıştır. Söz edilen sezgisel algoritmalar ağırlıklı olarak biyoloji, zooloji, fizik, bilgisayar ve karar üretme bilimlerinden türetilmiştir (Karaboğa, 2004).

Metasezgisel algoritmalar optimum çözüme ulaşma yolunda komşuluk tabanlı ve popülasyon tabanlı olmak üzere ikiye ayrılırlar. Komşuluk tabanlı algoritmalarda arama yapılırken başlangıçta rastgele ya da önceden belirlenmiş bir başlangıç çözümü ele alınır. Bu çözüm için komşuluklar belirlenir ve komşu çözümlerin performansları değerlendirilerek mevcut çözüm iyileştirilmeye çalışılır. Durdurma kriteri sağlanana kadar bu işlem devam eder. Böylelikle optimum nokta belirlenmiş olur. Tabu araştırma algoritması ve tavlama benzetimi algoritması komşuluk tabanlı metasezgisel algoritmalara örnek verilebilir. Popülasyon tabanlı algoritmalarda ise tek bir çözüm yerine aynı anda birden fazla çözüm ele alınarak, bu çözümlerin özelliklerine göre yeni daha iyi çözümler üretilir (Deb, 2005). Popülasyon tabanlı metasezgisel algoritmalara parçacık sürü algoritması (PSO), karınca kolonisi algoritması (KKA), diferansiyel gelişim algoritması (DGA), genetik algoritma (GA) ve son zamanlarda geliştirilen yerçekimsel arama algoritması (YAA) gibi algoritmalar örnek olarak verilebilir.

Genetik algoritma, özellikle doğrusal olmayan, çok değişkenli, zor problemlerin çözümüne yönelik olarak geliştirilmiş, popülasyon temelli sezgisel bir yöntemdir (Goldberg, 1989; Michalewicz, 1992; Reeves, 1995). Önbilgi ve varsayımlar olmadan, sadece amaç fonksiyonu ile çalışabilmektedir (Keskintürk ve Şahin, 2009).

GA, rassal arama tekniklerini kullanarak çözüm bulmaya çalışan, parametre kodlama esasına dayanan bir arama tekniğidir. Genetik algoritmalar mümkün olan çözümlerin bir popülasyonu üzerinde işlem yapan olasılıklı (stokastik) arama

algoritmalarıdır (Shapiro, 2001). Geleneksel programlama teknikleriyle çözülmesi güç olan, özellikle sınıflandırma ve çok boyutlu optimizasyon problemleri, bunların yardımıyla daha kolay ve hızlı olarak çözüme ulaştırılmaktadır. Genetik algoritmalar doğada geçerli olan en iyinin yaşaması kuralına dayanarak sürekli iyileşen çözümler üretir. Bunun için "iyi"nin ne olduğunu belirleyen bir uygunluk fonksiyonu ve yeni çözümler üretmek için çaprazlama, mutasyon gibi operatörleri kullanır. En iyi çözümü elde edebilmek için, zayıf çözümleri evrimsel bir işleyişe göre eleme yoluna gider. En iyi çözüm yapılan çevrimler sonucunda hala hayatta kalabilmeyi başaran çözümdür (Mitchell, 1998). Genetik algoritmalar, en iyinin korunumu ve doğal seçilim ilkesine dayanarak, benzetim yoluyla bilgisayarlara uygulanan ve bilgisayar üzerinde oluşan bir evrim şeklidir. Bu metot uzun çalışmaların neticesinde ilk defa John Holland tarafından uygulanmıştır (Beasley vd.., 1993).

Karınca kolonisi algoritması Marco Dorigo ve arkadaşları tarafından 1991'de önerilmiş metasezgisel bir algoritmadır. Algoritma gerçek karınca kolonilerinin davranışlarından esinlenerek geliştirilmiştir. Doğada karıncalar yiyeceğe giden en kısa mesafeyi keşfedebilirler. Bu yetenek aralarında oluşan iletişim ile gelişir. Karıncalar birbirleri ile feromen adı verilen kimyasal bir salgıyla iletişirler. Yol seçimi yaparken feromen miktarının fazla olduğu yolu tercih ederler. Dolayısıyla en kısa yolda feromen miktarı daha fazladır (Dorigo ve Di Caro, 1999). Gerçek karıncaların bu davranışları göz önüne alınarak yapay karıncalar ile karınca kolonisi algoritması geliştirilmiştir. Gerçek karınca kolonileri ile metaheuristik karınca kolonisi algoritması arasında işbirliği yapan bireylerden oluşan bir koloni olması, bireylerin koku bırakmaları, en kısa yolu bulma arayışları ve stokastik durum geçişleri bakımından benzerlik gösterirler (Dorigo vd., 1999). Farklı oldukları noktalar ise, yapay karıncaların ayrık bir dünyada yaşamaları, hafızalarının olması ve depoladıkları feromenin fonksiyondan elde ettikleri çözüm kalitesine göre olmasıdır. Algoritma özellikle en kısa yol problemleri olmak üzere birçok kombinatoryel problemlerde kullanılmış ve iyi sonuçlar elde edilmiştir (Alaykıran ve Engin, 2005).

Algoritma tüm yapay karıncalar için yolların üretilmesiyle başlar. Bir sonraki adımda üretilen yapay yolların uzunluğu hesaplanır. Bu yolların üzerinde bulunan feromen maddesi miktarının güncellenmesiyle devam eder. Son olarak, o ana kadar

bulunan en kısa yapay yolun hafızada tutulmasıyla algoritma, sonlandırma kriterine rastlayana kadar devam eder. KKA gezgin satıcı problemleri, karesel atama problemleri, ağ rotalama, araç rotalama problemleri, atölye tipi çizelgeleme problemleri ve maksimal kısıt sağlama problemleri gibi alanlarda uygulanmıştır.

Diferansiyel gelişim algoritması, Storn ve Price tarafından 1995'te geliştirilmiş popülasyon temelli bir sezgisel optimizasyon tekniğidir (Storn ve Price, 1995). Gerçek değerli parametrelere sahip problemlerin çözümünde küresel optimizasyonlarda kullanılmak üzere tasarlanmış basit bir algoritmadır. İşleyiş ve operatörleri sebebiyle genetik algoritmaya benzer (Keskintürk, 2006). Genetik algoritmada yer alan mutasyon, çaprazlama ve seleksiyon işlemleri DGA'da da yer almaktadır. Farklı olarak bu işlemler yenilenecek her jenerasyonda tüm kromozomlara uygulanır. Daha sonra elde edilen yeni bireyin uygunluğuna bakılarak seçim gerçekleştirilir.

Optimize edilecek olan fonksiyonun değişken sayısı, DGA'da gen sayısıdır. D adet genin oluşturduğu vektöre kromozom adı verilir. Kromozomların sayısı (NP) 3'ten büyük alınmalıdır. Çünkü algoritma, işlem yapılacak kromozom dışında 3 adet daha kromozoma ihtiyaç duyar. Algoritma rastgele çözümlerle başlatılır. Oluşturulan popülasyondaki her kromozomun uygunluk değeri hesaplanır. Bu kromozomlar arasından birbirinden ve geliştirilecek vektörden farklı 3 kromozom seçilir. Seçilen kromozomlardan ilk ikisinin farkı alınarak ölçekleme faktörü ile çarpılır. Yapılan çalışmalarda ölçekleme faktörü [0,2] aralığında alınmıştır. Bu fark vektörü seçilen diğer kromozom ile toplanarak aday vektör elde edilir. Popülâsyonda bulunan tüm diğer vektörler için de aday vektörler oluşturulur. Böylelikle mutasyon ve rekombinasyon işlemleri gerçekleştirilmiş olunur. Elde edilen vektörler ile mevcut vektörler karşılaştırılarak seçim işlemi gerçekleştirilir. Algoritma iterasyon sayısı boyunca devam ettirilir. Algoritma tamamlandığında popülâsyondaki en iyi kromozom çözüm olarak bulunmuş olur. DGA'nın birçok uygulama alanı vardır. Gezgin satıcı problemleri, imge bölütleme, filtre tasarımı, yapay ağ tasarımı, bulanık mantık kontrolcü dizaynı ve sistem kimliklendirme kullanıldığı alanlara örnek verilebilir.

Parçacık sürü optimizasyonu, Eberhart ve Kennedy tarafından 1995'te, kuş ve balık gibi sürülerin davranışlarından esinlenerek tasarlanan bir optimizasyon yöntemidir (Eberhart ve Kennedy, 1995). Algoritma rastgele çözümlerden oluşan bir popülâsyonla

başlatılır. Popülâsyonu oluşturan olası çözümlere parçacık adı verilir. Arama uzayına rastgele dağıtılan parçacıklara, uzayın sınırlılıklarına göre rastgele hızlar atanır. Belli bir hıza sahip olan parçacıklar kendilerinin ve sistemin şu ana kadar gerçekleştirdiği en iyi çözüme doğru uçarlar. Bu olay da her parçacığın kendi rotasını hafızasında tuttuğu manasına gelir. Her iterasyonda parçacıkların uygunluk değerleri hesaplanır. Uygunluk fonksiyonundan elde edilen sonuçlar ile parçacığın hafızasında olan sonuçlar karşılaştırılır. Karşılaştırma sonrasında o ana kadar mevcut olan en iyi sonuç elde edilir. Parçacığın elde ettiği bu sonuç yerel en iyi (pbest) olarak isimlendirilir. Sistemdeki tüm parçacıkların o ana kadar elde ettiği en iyi sonuca ise küresel en iyi (gbest) adı verilir. Gerçeklenen her iterasyonda parçacıklar için pbest ve gbest değerlerine göre hızları belirlenir. Böylelikle her iterasyonda parçacıkların ilgili uzayda konumları güncellenerek sonlandırma kriteri sağlandığında sistemdeki en iyi çözüm bulunmuş olur (Eberhart ve Shi, 2001).

PSO az sayıda parametre kullanımı, az miktarda hafızaya gereksinim duyma, hızlı hesaplama ve yakınsama gibi avantajlarından dolayı optimizasyon problemlerinde sıklıkla kullanılmıştır (Erdoğmuş, 2010). Bunlardan bazılarına sinir ağları (neural network) eğitimi, güç ve voltaj kontrolü, sipariş miktarı belirleme (lot sizing) problemi, akış tipi çizelgeleme problemleri, gezgin satıcı problemi, iş atama problemi örnek olarak verilebilir. Ayrıca PSO, çok boyutlu doğrusal olmayan fonksiyonların optimizasyonu, bulanık sistem kontrolü gibi birçok alanda da başarıyla uygulanabilmektedir.

Bu tez çalışmasında optimizasyon yöntemleri incelenerek, metasezgisel optimizasyon yöntemlerinden olan YAA'dan esinlenilmiş ve iki adet geliştirilmiş YAA algoritması türetilmiştir. Böylelikle optimizasyon problemlerinde daha etkili algoritmalar kullanılarak, gerçek değerlere daha yakın çözümler elde edilebilecektir.

2. YERÇEKİMSEL ARAMA ALGORİTMASI (YAA)

Yerçekimsel arama algoritması (YAA), Rashedi ve arkadaşları (2009) tarafından bulunan Newton'un hareket ve yerçekimi kanunlarına dayalı bir optimizasyon algoritmasıdır. Bu nedenle sezgisel optimizasyon yöntemlerinden biri olan YAA'yı ele almadan önce Newton'un hareket ve yerçekimi kanunlarına yer verilmelidir.

2.1. Newton'un Hareket Kanunları

2.1.1. Eylemsizlik yasası

Evrendeki herhangi bir cisim üzerine hiçbir kuvvet etki etmiyorsa ya da cisme etki eden kuvvetlerin vektörel olarak toplamı sıfıra eşitse, eğer cisim belli bir hızda hareket ediyorsa düzgün doğrusal hareketine devam eder, duruyorsa durağanlığını korur. Dolayısıyla bu yasada cismin ivmesinin sıfır olması söz konusudur (Serway ve Beichner, 2007).

2.1.2. İvme yasası

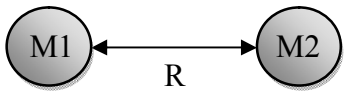
Bir cisim üzerine etki eden kuvvetlerin vektörel olarak toplamı sıfıra eşit değilse yani cisim herhangi bir kuvvet yönünde çekiliyorsa, cisimde bir hız değişimi olur. Bu hız değişimine ivme adı verilir. Cismi hareket ettiren ya da hızlandıran kuvvet ile cismin ivmesi aynı doğrultudadır. Herhangi iki cisme aynı yönde aynı büyüklükte kuvvet uygulandığında kütlesi büyük olanın ivmesi daha küçük, kütlesi küçük olan cismin ivmesinin ise daha büyük olduğu görülmüştür. Yani bir cismin ivmesi, üzerine uygulanan kuvvet ile doğru, kütlesi ile ters orantılıdır. Bu oran $F = m \times a$ şeklinde gösterilmektedir (Serway ve Beichner, 2007; Rashedi vd., 2009).

2.1.3. Etki-tepki yasası

İki cisim arasında herhangi bir kuvvetle oluşan etki her zaman kendisine eşit büyüklükte, aynı doğrultuda fakat ters yönde bir tepkiye neden olur (Serway ve Beichner, 2007).

2.2. Newton'un Evrensel Çekim Kanunu

Evrendeki cisimler birbirlerini yerçekimine bağlı olarak çekerler. Bu kütle çekim kuvveti, cisimlerin kütlelerinin çarpımıyla doğru, cisimler arasındaki uzaklıkla ters orantılıdır. Şekil 2.1.'de birbirine etki eden M1 ve M2 noktasal kütleleri görülmektedir. Bu kütleler arasındaki uzaklık R ile gösterilmiştir.



Şekil 2.1. Birbirine etki eden kütleler.

Newton'un yerçekimi kanununa göre kütleler arasındaki kuvvet:

$$F = G \times \frac{M1 \times M2}{R^2} \quad \text{(E.2.1)}$$

ile ifade edilir. Burada G yerçekimi sabitidir (Serway ve Beichner, 2007; Rashedi vd., 2009).

2.3. Yerçekimine Dayalı YAA

YAA, Newton'un hareket kanunlarından ikincisi olan ivme kanunu ve yine Newton'un evrensel çekim kanunu temel alınarak tasarlanmış bir optimizasyon yöntemidir. YAA'da arama uzayındaki her bir parçacık bir kütle olarak kabul edilir. Bu nedenle YAA'yı bir yapay kütle sistemi olarak ifade etmek mümkündür (Rashedi vd., 2009). Arama uzayındaki tüm kütleler Newton'un evrensel çekim kanununa göre birbirlerini çekerler ve yerçekimi kuvveti ile birbirlerine kuvvet uygularlar. Bu kuvvete maruz kalan kütleler arama uzayı içerisinde hareket ederek optimum çözüme ulaşırlar. Noktasal kütlelerin bu şekildeki hareketi Newton'un ikinci hareket kanununa uygun olarak meydana gelir. Arama uzayındaki her bir kütlenin; konumu, eylemsizlik kütlesi, aktif yerçekimsel kütlesi ve pasif yerçekimsel kütlesi olmak üzere dört özelliği vardır.

Her kütle arama uzayında belli bir pozisyonda bulunur. Bulundukları konum, çözülmesi beklenen problem için birer çözüm alternatifidir. Kütleler, üzerine herhangi bir kuvvet etki ettiğinde, bu kuvvete karşı koymak isteyerek bir direnç gösterir. Bu dirence eylemsizlik kütlesi denir. Eylemsizlik kütlesi fazla olan kütle daha yavaş hareket etmek isteyecek, az olanı ise daha hızlı hareket edecektir. Belli bir obje nedeniyle oluşan yerçekimi alanı gücüne aktif yerçekimsel kütle denir. Yerçekimi alanı

ile bir nesnenin iletişim gücüne pasif yerçekimsel kütle denir (Chatterjee ve Mahanti, 2010).

YAA' da her bir nesnenin kütle miktarı o nesnenin performansını gösterir. Her bir kütle, arama uzayında olan diğer kütleleri yerçekimi kuvveti ile çeker. Böylelikle kütleler arası etkileşim sağlanmış olur. Bu kuvvet bütün kütlelerin en ağır olan kütleye doğru hareket etmesini sağlar. Bundan dolayı da kütleler yerçekimi kuvveti doğrultusunda birlikte hareket ederler. Algoritma boyunca en ağır olan kütle diğer kütlelere nazaran daha yavaş hareket edecek ve diğerlerini kendine çekecektir. İterasyon sayısı bitiminde ya da herhangi bir sonlandırma eylemi olduğunda kütlesi en fazla olan nesne, problemin optimum çözümünü oluşturmuş olacaktır.

2.4. Algoritma Adımları

Algoritma adımları Rashedi ve arkadaşlarının (2009) yaptıkları çalışma baz alınarak aşağıdaki gibi verilebilir.

2.4.1. Başlangıç değerlerinin atanması

YAA'da kullanılan yerçekimi sabitinin (G), belli bir iterasyon sonunda optimizasyon tamamlanacaksa maksimum iterasyon sayısının, ve bir kütlenin diğer tüm kütlelere yapacağı etkiyi hesaplamada kullanılan küçük bir sabitin (ε) ilk değerlerinin atandığı adımdır (Ceylan vd., 2010).

2.4.2. Arama uzayının tanımlanması

Problemlerin optimize edilebilmesi için olası sonuçların belirlenmesi gerekmektedir. Bu çözümler belli bir arama uzayı içerisinde tanımlanır. YAA için her mümkün çözüm bir kütledir. Sistemin kaç kütleden oluşacağı bu bölümde tasarlanır. Oluşturulan kütleler arama uzayına rastgele yerleştirilir. N adet kütleden oluşan bir arama uzayında i. kütlenin pozisyonu:

$$X_i = \left(x_i^1, \dots, x_i^d, \dots, x_i^n\right) \qquad i = 1,2, \dots, N \qquad \text{(E.2.2)}$$

şeklinde belirlenir. Burada x_i^d, i. kütlenin d. boyuttaki konumunu gösterir.

2.4.3. Yerçekimi sabitinin hesaplanması

Algoritmanın ilk kısmında yerçekimi sabitine verilen değerin her iterasyonda azaltılması gerekmektedir. Zamanla azalan yerçekimi sabiti ile arama hızı kontrol edilmektedir. t anındaki yerçekimi sabiti

$$G(t) = G_0 \times e^{\left(-\propto \frac{t}{T}\right)} \quad \text{(E.2.3)}$$

eşitliği ile hesaplanır. Burada G_0, yerçekimi sabitinin başlangıç değerini, α, kullanıcının belirlediği sabit bir değeri, t, o andaki iterasyon sayısını ve T, maksimum iterasyon sayısını gösterir.

2.4.4. Uygunluk fonksiyonu ile uygunluk değerlerinin hesaplanması

Her bir kütlenin o andaki uygunluk değeri uygunluk fonksiyonu ile hesaplanır. Hesaplanan uygunluk değerlerinin en iyi ve en kötüleri seçilir. Eğer bir problem minimize edilmek isteniyorsa;

$$best(t) = \min_{j=\{1,\ldots,N\}} fit_j(t) \quad \text{(E.2.4)}$$

$$worst(t) = \max_{j=\{1,\ldots,N\}} fit_j(t) \quad \text{(E.2.5)}$$

Maksimizasyon yapılmak isteniyorsa;

$$best(t) = \max_{j=\{1,\ldots,N\}} fit_j(t) \quad \text{(E.2.6)}$$

$$worst(t) = \min_{j=\{1,\ldots,N\}} fit_j(t) \quad \text{(E.2.7)}$$

eşitlikleri kullanılır. $fit_j(t)$, j. kütlenin t anındaki uygunluk değeri, $best(t)$, t anındaki en iyi çözüm, $worst(t)$ ise t anındaki en kötü çözüme karşılık gelir.

2.4.5. Kütle hesabı

Arama uzayında bulunan bir kütlenin aktif yerçekimsel kütlesi, pasif yerçekimsel kütlesi ve eylemsizlik kütlesi birbirlerine eşit alınarak, tüm kütleler hesaplanır.

$$M_{ai} = M_{pi} = M_{ii} = M_i, \qquad i = 1,2,\ldots,N \quad \text{(E.2.8)}$$

$$m_i(t) = \frac{fit_i(t) - worst(t)}{best(t) - worst(t)} \quad \text{(E.2.9)}$$

$$M_i(t) = \frac{m_i(t)}{\sum_{j=1}^{N} m_j(t)} \quad \text{(E.2.10)}$$

kütlenin aktif yerçekimsel kütlesi M_{ai}, pasif yerçekimsel kütlesi M_{pi}, eylemsizlik kütlesi M_i olmak üzere öncelikle $m_i(t)$ yani t anında i. kütlenin değeri hesaplanmaktadır. E.2.10.'da ise normalizasyon işlemi gerçekleştirilmektedir. Kütlelerin güncelleştirilmesiyle belirlenen en ağır kütle en etkili olanıdır. Arama uzayındaki daha etkili olan kütle diğerlerine göre daha yavaş hareket eder ve diğerlerini daha iyi çeker (Chatterjee vd., 2011).

2.4.6. Kuvvet hesabı

E.2.1.'de iki kütlenin birbirine uyguladığı kuvvet gösterilmiştir. Newton'un evrensel çekim kanununda kütleler arası mesafenin karesinin, kütlelerin çarpımına bölündüğü bilinmektedir. Rashedi ve arkadaşları (2009) yaptıkları çalışmada deneysel sonuçları göz önüne alarak evrensel çekim yasasında kullanılan ve R^2olarak gösterilen iki kütle arasındaki mesafenin karesi yerine R kullanmışlardır. Arama uzayındaki iki noktasal kütlenin arasındaki Öklit mesafesi;

$$R_{ij}(t) = \left\| X_i(t), X_j(t) \right\|_2 \quad \text{(E.2.11)}$$

olarak hesaplanmıştır. Bu eşitlikte $R_{ij}(t)$ i ve j kütlelerinin arasındaki mesafeyi vermektedir. Dolayısıyla bu kütlelerin arasındaki kuvvet;

$$F_{ij}^{d}(t) = G(t) \times \frac{M_{pi}(t) \times M_{aj}(t)}{R_{ij}(t) + \varepsilon} \times \left(X_j^{d}(t) - X_i^{d}(t) \right) \quad \text{(E.2.12)}$$

eşitliği ile hesaplanır. $F_{ij}^{d}(t)$, t anında d. boyutta i ve j kütleleri arasındaki kuvveti, $G(t)$ yerçekimi sabitini, $M_{pi}(t)$ t anında i. kütlenin pasif yerçekimsel kütlesini, $M_{aj}(t)$ t anında j. kütlenin aktif yerçekimsel kütlesini, ε kullanıcı tarafından karar verilen sayısal olarak ufak bir sabiti $X_j^{d}(t)$ ve $X_i^{d}(t)$ ise i ve j kütlelerinin d. boyuttaki konumunu belirtir.

İki kütle arasındaki kuvvet hesaplandıktan sonra bir kütleye etkiyen toplam kuvvet hesaplanır:

$$F_i^{d}(t) = \sum_{j \in Kbest, j \neq i}^{N} rand_j F_{ij}^{d}(t) \quad \text{(E.2.13)}$$

E.2.13.'de $\mathrm{rand_j}$ [0,1] aralığında verilen rastgele bir sayıdır. Kbest ise başlangıçta K_0 kadar kütle ile başlayarak zamanla lineer olarak azaltılan bir değerdir. Newton'un evrensel çekim kanununa göre arama uzayındaki objelerin birbirlerini çektiği bilinmektedir. Kbest değeri ile arama uzayı içerisinde belirlenen en kötü objenin diğer tüm objelere etkidiği kuvvetler sıfırlanır. Daha sonraki iterasyonlarda bağlantısı kesilen obje, diğer objeleri kendine çekemeyecek, sadece diğerleri tarafından çekime girebilecektir. Bu nedenle E.2.13. ile i. obje üzerine etkiyen toplam kuvvet hesaplanırken, j. üye Kbest üyelerinden biri olması gerekmektedir. Algoritmanın sonunda Kbest'in son değeri bire kadar düşer. Böylelikle sadece algoritma sonunda elde edilen en iyi çözüm, arama uzayındaki diğer objeleri kendine çeker, diğer objeler en iyiye bir kuvvet uygulayamaz.

2.4.7. İvme hesabı

5. adımda kütle ve 6. adımda kuvvet hesabı sonucunda Newton'un ikinci yasası olan ivme yasasına dayanarak kütlelerin ivmeleri hesaplanır;

$$\mathrm{a_i^d(t)} = \frac{\mathrm{F_i^d(t)}}{\mathrm{M_{ii}(t)}} \qquad \text{(E.2.14)}$$

$\mathrm{a_i^d(t)}$ d. boyuttaki i. kütlenin t anındaki ivmesini verir.

2.4.8. Hız ve konum güncellemesi

Arama uzayındaki tüm kütleler etkileşime girerek birbirlerine ivme kazandırırlar. Bu nedenle bir iterasyon sonraki hız E.2.15.'de de görüldüğü üzere kütlenin o andaki hızı ile o anda oluşan hız değişiminin toplamına eşittir.

$$\mathrm{V_i^d(t+1)} = \mathrm{rand_i} \times \mathrm{V_i^d(t)} + \mathrm{a_i^d(t)} \qquad \text{(E.2.15)}$$

$\mathrm{rand_i}$ [0,1] aralığında rastgele atanan bir sayıdır.

Hız değişimiyle birlikte her bir kütlenin sistemdeki yeri de değişmektedir. Bu yüzden E.2.16. ile konum güncellemesi yapılmaktadır (Taghipour vd., 2010).

$$\mathrm{X_i^d(t+1)} = \mathrm{X_i^d(t)} + \mathrm{V_i^d(t+1)} \qquad \text{(E.2.16)}$$

2.4.9. Sonlandırma

Herhangi bir sonlandırma kriteri olduğunda algoritma durdurulur, yoksa algoritmanın yerçekimi sabitinin güncellendiği 3. adımına geri dönülerek sonlandırma kriterine rastlayana kadar algoritma devam ettirilir.

2.5. Test Fonksiyonları

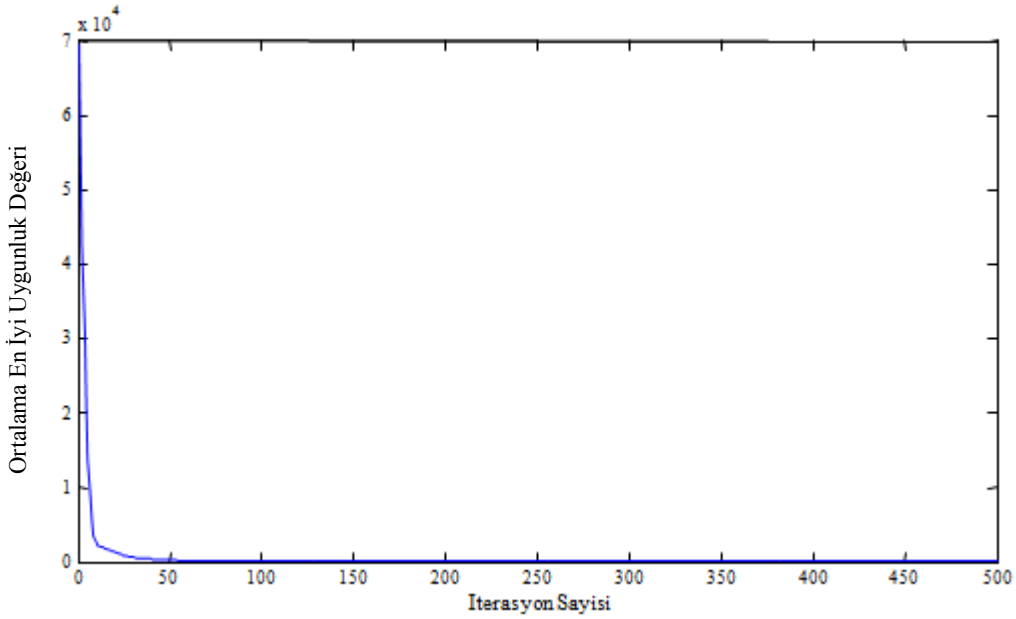
Sphere, Ackley ve Humpcamel test fonksiyonları kullanılarak YAA'nın çalışması değerlendirilmiştir. YAA kodları ve fonksiyonlar MATLAB programlama dilinde yazılmış ve Intel(R) Core(TM) i5 CPU M 430 @ 2.27 GHz, 4GB RAM donanımlı, 32 bit işletim sistemine ait bir dizüstü bilgisayarda çalıştırılmıştır.

2.5.1. Sphere fonksiyonu

Test fonksiyonlarından ilki olan Sphere fonksiyonu tek bir global minimuma sahip bir fonksiyondur. Bu tür fonksiyonlara tek modlu fonksiyonlar denir. Sphere fonksiyonunun ayrıca düzgün ve ileri derecede konveks bir fonksiyon olma özelliği vardır.

$$F_{\text{Sphere}}(X) = \sum_{i=1}^{n} x_i^2 \qquad \text{(E.2.17)}$$

E.2.17.'de verilen Sphere fonksiyonunun global minimum noktası 0'a eşittir. Eşitlikte n, fonksiyonun boyutunu vermektedir. Yaptığımız çalışmada YAA kullanılarak Sphere fonksiyonunun global minimumuna yakınsanmıştır. YAA'da popülasyon büyüklüğü 25, fonksiyon boyutu 30, maksimum iterasyon sayısı 500, değişken sınırları [-100,100], $\varepsilon = 10^{-12}$, yerçekimi sabitinin hesaplamasında α = 10, G_0= 50 ve son olarak K popülasyon büyüklüğünün değeri kadar alınıp, her iterasyonda lineer olarak azaltılmıştır. Şekil 2.2.'de 500 iterasyonun sonucunda en iyi uygunluk 0.000121692999 olarak bulunmuş ve toplam hesaplama süresi 8.778853 sn olarak ölçülmüştür.



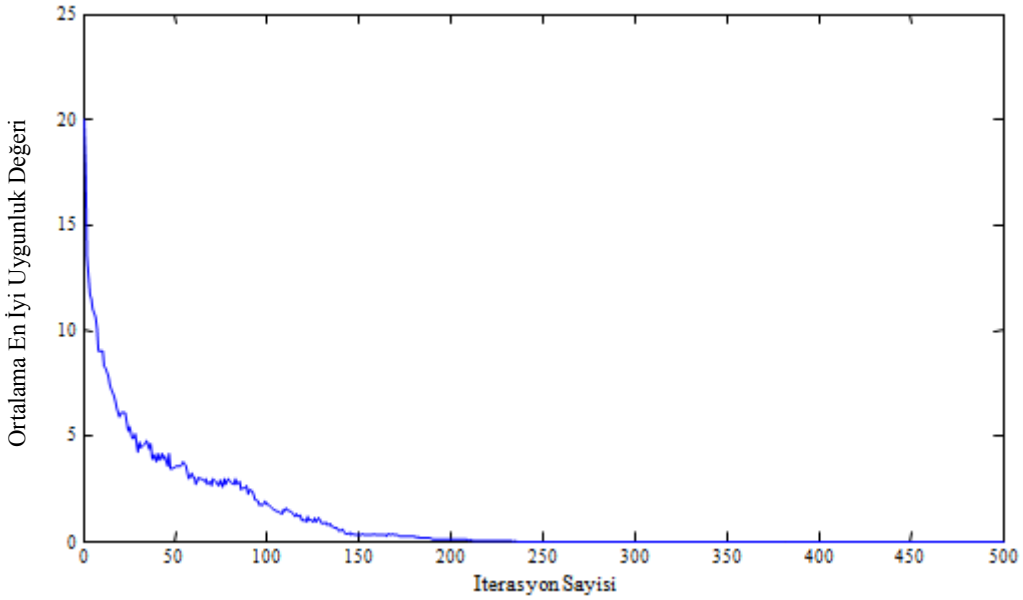
Şekil 2.2. YAA'nın 500 iterasyon boyunca Sphere fonksiyonunda ürettiği uygunluklar.

2.5.2. Ackley fonksiyonu

Ackley fonksiyonu birçok yerel minimuma ancak tek bir global minimuma sahip bir fonksiyondur. Bu tür fonksiyonlara çok modlu fonksiyonlar denir.

$$F_{Ackley}(X) = -20 \times \exp\left(-0.2 \times \sqrt{\frac{\sum_{i=1}^{n} x_i^2}{n}}\right) - \exp\left(\frac{\sum_{i=1}^{n} \cos(2\times\pi\times x_i)}{n}\right) + 20 + e \quad \text{(E.2.18)}$$

E.2.18.'de verilen Ackley fonksiyonunun global minimum noktası 0'a eşittir. Eşitlikte n, fonksiyonun boyutunu vermektedir. YAA'da popülasyon büyüklüğü 25, fonksiyon boyutu 30, maksimum iterasyon sayısı 500, değişken sınırları [-32,32], $\varepsilon = 10^{-12}$, yerçekimi sabiti hesabında α = 10, G_0= 50 ve son olarak K popülasyon büyüklüğünün değeri kadar alınıp, her iterasyonda lineer olarak azaltılmıştır. Şekil 2.3.'de maksimum iterasyon sonrası global minimum noktası 0.004274278078 olarak bulunmuştur. Minimum uygunluk değerinin bulunması için geçen süre ise 9.347089 sn'dir.



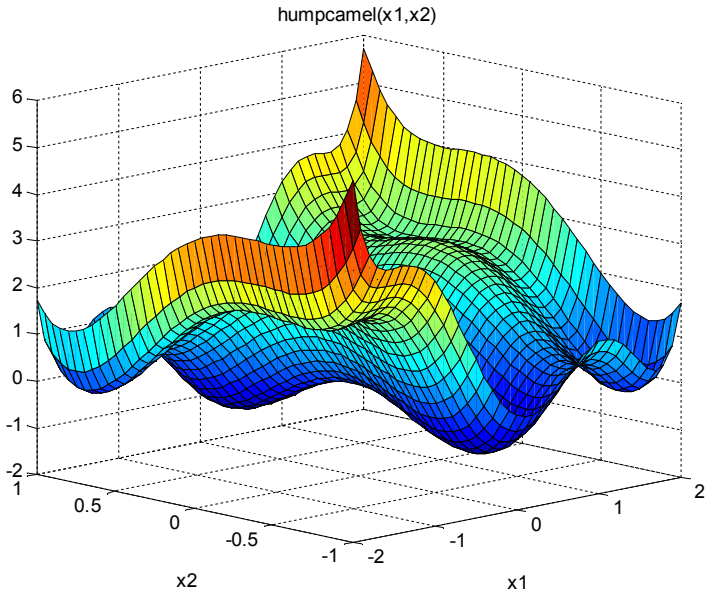
Şekil 2.3. YAA'nın 500 iterasyon boyunca Ackley fonksiyonunda ürettiği uygunluklar.

2.5.3. Humpcamel fonksiyonu

Humpcamel fonksiyonu 2 global minimumla birlikte toplamda 6 adet yerel minimuma sahiptir. Çok modlu bir fonksiyondur.

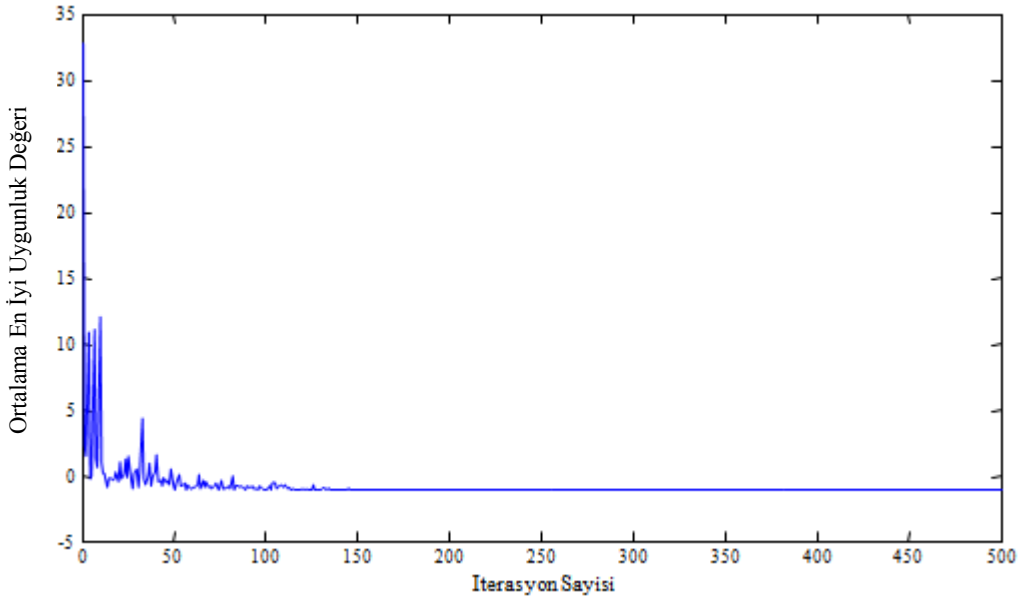
$$F_{\text{Humpcamel}}(X) = 4 \times x_1^2 - 2.1 \times x_1^4 + \frac{x_1^6}{3} + x_1 \times x_2 - 4 \times x_2^2 + 4 \times x_2^4 \qquad \text{(E.2.19)}$$

Humpcamel fonksiyonu E.2.19.'da verilmiş olup, x_1 ve x_2 değişkenlerinden oluştuğu görülmektedir. Bu nedenle fonksiyon 2 boyutludur ve global minimum noktası -1.0316'dır. Şekil 2.4.'de Six Humpcamel fonksiyonu gösterilmektedir.



Şekil 2.4. Six Humpcamel fonksiyonu.

YAA'da popülasyon büyüklüğü 25, maksimum iterasyon sayısı 500, değişken sınırları [-5,5], $\varepsilon = 10^{-12}$, yerçekimi sabiti hesabında α = 10, G_0= 50 ve son olarak K popülasyon büyüklüğünün değeri kadar alınıp, her iterasyonda lineer olarak azaltılmıştır. Şekil 2.5.'de maksimum iterasyon sayısı sonucunda Humpcamel fonksiyonunun global minumum değeri -1.031627931600 olarak bulunup, hesaplama süresi 8.761347 sn'dir.



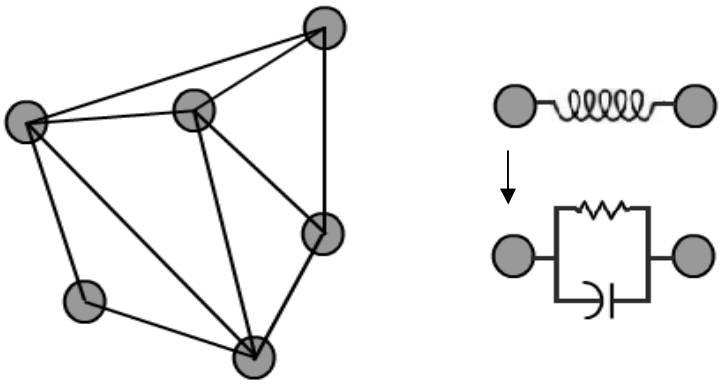
Şekil 2.5. YAA'nın 500 iterasyon boyunca Humpcamel fonksiyonunda ürettiği uygunluklar.

3. KÜTLE-YAY SİSTEMİ

Günümüzde görselliğin ön plana çıkmasıyla birlikte bilgisayar grafiklerinde birçok gelişme yaşanmıştır. Modelleme ve simülasyon yaşamın çoğu alanında uygulanmaya başlanmıştır. Araştırmacılar gerçeğe yakın ve hızlı sonuç veren yöntemler geliştirme noktasında çalışmalar yapmıştır. Bu yöntemler fiziksel ve fiziksel tabanlı olmayan yöntemler olmak üzere ikiye ayrılır (Duysak, 2005). Modelleme ve simülasyonda özellikle fiziksel tabanlı yöntemlerden olan SEM (Sonlu elemanlar yöntemi) ve MSS (kütle-yay sistemi) yöntemlerinin uygulandığı görülmüştür. Çalışmalar incelendiğinde hassasiyetin önemli olduğu durumlarda SEM, hızın önemli olduğu durumlarda ise MSS tercih edilmiştir (Duysak ve Zhang, 2005). MSS'nin kullanıldığı uygulamalara yumuşak doku deformasyonu, saç ve kumaş simülasyonları örnek olarak verilebilir.

3.1. Kütle-Yay Yapısı

Bir nesnenin 3 boyutlu modellemesi yapılırken n adet düğüm ve bu düğümleri anlamlı olarak birbirine bağlayan m adet bağlantı kullanılır. Düğüm ve bağlantılardan oluşan bu ağsal yapı, üçgenlerle ya da çokgenlerle ifade edilebilir. Şekil 3.1.'de bir nesnenin üçgenlerden oluşan ağ modelinin bir kesiti görülmektedir. Kütle yay sisteminde nesneyi oluşturan her düğüm, noktasal kütle olarak adlandırılmaktadır. Genellikle noktasal kütleler, modellenen nesnenin toplam kütlesinin, noktasal kütle sayısına bölümüyle elde edilir.



Şekil 3.1. Kütle-yay yapısının gösterimi.

Objeye dışarıdan herhangi büyüklükte ve yönde kuvvet etki ettiğinde noktasal kütleler o kuvvetin yönüne doğru ve şiddetine bağlı olarak yer değiştirir. Uygulanan kuvvet dolayısıyla, kütlelerin yer değişimi yaylarda gerilmeye veya sıkışmaya neden olur. Böylelikle modeli oluşturan ağsal yapıda, kütleler arası etkileşim meydana gelir.

Kütle-yay sistemi, ağda mevcut olan kütle ve yayların birbirleriyle ve kendi içindeki değişimlerini inceleyen bir yöntemdir. Uygulanan kuvvetin büyüklüğüne ve yönüne göre, etki ettiği noktadan itibaren nesne üzerindeki değişimler deformasyonu ifade eder.

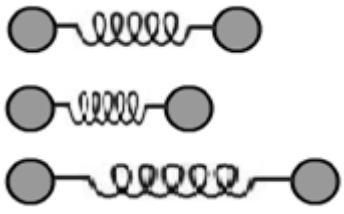
3.2. Yay Dinamiği

İki noktasal kütle arasındaki bağlantıya yay adı verilir. Yaylar belli bir ağırlığa sahip değildir ve kütleler arasındaki iletişimi sağlar. Yayların kütleler arası iletişimi sağlamasındaki rolü, deformasyon karakteristiklerinin yay parametreleri tarafından ağsal yapıya gömülmesi olarak açıklanabilir. Yay parametreleri sertlik ve sönümlemedir (Şekercioğlu, 2010).

Yaya bağlı noktasal kütlelerden herhangi birinde yer değiştirme söz konusu olduğunda yay dinamiği süreci başlar. Kütlesel noktada olan herhangi bir değişim ya da dışarıdan bir kuvvet, yayın içsel bir kuvvet oluşturmasına neden olur. Yayın simülasyon sırasında davranışını belirlemek için bazı özellikleri kullanılır (Mahal vd., 2001). Bunlar:

- Yayın model içindeki konumu,
- Yayın sertliği,
- Yayın değişimden önceki uzunluğudur.

Şekil 3.2.'de iki kütle arasındaki yay, kuvvetin yaptığı etkiyle, gerilir ya da sıkışır. Böylelikle yayın başlangıç uzunluğu azalır ya da artar.



Şekil 3.2. Yayların gerilmesi ve sıkışması.

Yayın ilk uzunluğunun herhangi bir etkiyle değişmesi sonucu yay bir kuvvet oluşturur. Bu kuvvet Hooke kanununa göre aşağıdaki eşitlikte gösterilmiştir:

$$\mathrm{f} = -\mathrm{k_s} \times (|\mathrm{d}| - \mathrm{s}) \times \frac{\mathrm{d}}{|\mathrm{d}|} \qquad \text{(E.3.1)}$$

Yayın bağladığı iki noktasal kütlenin konumları arasındaki fark, yayın uzunluğunu ifade eder. E.3.1.'de s, yaya herhangi bir etki etmeden önceki yani ilk uzunluğudur, $|d|$ ise yayın işlem yapılacağı andaki uzunluğunu belirtir. k_s ile gösterilen değer ise yayın sertliğini gösteren yay sabitidir. Eşitliğin son kısmında bulunan $\frac{d}{|d|}$ birim vektörü ifade eder.

Eğer sistem bu şekilde bırakılırsa, kütle-yay sisteminde titreşim yaşanmaya başlar. Bu nedenle sönümleme E.3.1.'e ekilmelidir.

$$\mathrm{f} = -\left(\mathrm{k_s} \times (|\mathrm{d}| - \mathrm{s}) + \mathrm{k_d} \times \frac{\mathrm{d} \cdot \mathrm{d}}{|\mathrm{d}|}\right) \times \frac{\mathrm{d}}{|\mathrm{d}|} \qquad \text{(E.3.2)}$$

Toplam yay kuvveti yukarıdaki eşitlikte gösterilmiştir. Burada k_d sönümleme sabiti, d iki noktasal kütlenin konum farkı ve hız farkını, $\mathrm{d} \cdot \mathrm{d}$ ifadesi ise iki vektörel büyüklüğün çarpımını ifade etmektedir (Duysak vd., 2003).

3.3. Kütle Dinamiği

Dış kuvvetler veya yayların dinamiği sonucunda kütle dinamiği süreci başlamış olur. Kütleye etki eden toplam kuvvet belirlendikten sonra Newton'un ikinci kanununa göre ivme hesaplanır.

$$\mathrm{f} = \mathrm{m} \times \mathrm{a} \qquad \text{(E.3.3)}$$

İvmesi bilinen kütlenin hızı ve yeni konumu Euler integrasyonlarından biri seçilerek belirlenir.

3.3.1. Explicit Euler integrasyonu

Kütlenin bir sonraki hızı E.3.4. ile hesaplanır. Yeni konumu ise önceki konumu ile yer değiştirmesi toplanarak elde edilir (Provot, 1995).

$$\mathrm{V}(\mathrm{t} + \Delta\mathrm{t}) = \mathrm{V}(\mathrm{t}) + \Delta\mathrm{t} \times \frac{\mathrm{F}(\mathrm{t})}{\mathrm{m}} \qquad \text{(E.3.4)}$$

$$X(t + \Delta t) = X(t) + \Delta t \times V(t + \Delta t) \quad \text{(E.3.5)}$$

3.3.2. Implicit Euler integrasyonu

Explicit Euler integrasyonundan tek farkı bir sonraki hızın hesaplanmasında kullanılan kuvvetin seçimidir. Explicit Euler integrasyonunda kuvvet o andaki kuvvettir fakat İmplicit Euler integrasyonunda bir sonraki kuvvet ele alınır.

$$V(t + \Delta t) = V(t) + \Delta t \times \frac{F(t+\Delta t)}{m} \quad \text{(E.3.6)}$$

$$X(t + \Delta t) = X(t) + \Delta t \times V(t + \Delta t) \quad \text{(E.3.7)}$$

Kütle ve yaylardan oluşan modele dışarıdan bir kuvvet etki ettiğinde, yaylar sıkışarak ya da gerilerek kütlelere bir iç kuvvet uygular. Bu kuvvet ile birlikte kütleler bir ivmeye sahip olur. Dolayısıyla yer değiştirme söz konusudur. Kütle-yay sistemindeki bu etkileşim sonucu modelde oluşan değişim deformasyonu ifade eder. Kütle-yay sisteminin anlaşılabilirliğindeki kolaylık ve etkinliği açısından modelleme alanlarında geniş bir kullanıma sahiptir.

4. MSS-GSA ALGORİTMASI

Optimizasyon yöntemlerinde arama uzayına atanan objeler birbirinden bağımsız bir şekilde optimum noktayı ararlar. Birbirine bağlı olan objeler kullanıldığında optimum noktaya daha yakın bir çözüm elde etmek mümkündür. Önerilen algoritmada kütle yay sisteminin çalışma prensibi ele alınarak, optimum noktayı arama amaçlanmıştır. Bu nedenle YAA'da kullanılan kütlelerin arasına yay bağlanmıştır. Kütlelerin konum farkından dolayı potansiyel farktan doğan dış kuvvet ile bağlı olduğu yay tarafından oluşan iç kuvvet toplanarak bir üyeye uygulanan toplam kuvvet hesaplanmıştır. Böylelikle arama uzayında hem global arama hem de yerel arama yapılmıştır. Kütlelerin arama uzayı içerisindeki hareketi ise Newton'un ivme yasasına göre gerçekleştirilmiştir.

4.1. Algoritma Adımları

Algoritma adımları aşağıdaki gibi verilebilir.

4.1.1. İlk değerlerin atanması

Algoritma içerisinde kullanılan sabit değerler (ε ve $\propto$) , yerçekimi sabitinin başlangıç değeri (G_0), yay katsayısı (k), maksimum iterasyon sayısı, matematiksel olarak ifade edilen problemin boyutu ve optimizasyon sırasında kullanılacak üye sayısı bu adımda belirlenir.

4.1.2. Arama uzayının tasarlanması

Problemin sınırları göz önüne alınarak arama uzayının sınırları belirlenir. Oluşturulan uzaya, üyeler rastgele konumlandırılır.

$$X_i = \left(x_i^1, \dots, x_i^d, \dots, x_i^n\right) \qquad i = 1,2, \dots, N \qquad \text{(E.4.1)}$$

x_i^d, arama uzayındaki i. üyenin d. boyuttaki konumunu belirtir.

4.1.3. Üyelerin uygunluk değerlerinin hesaplanması

Üyeler uygunluk fonksiyonuna gönderilerek her birinin uygunluk değerleri hesaplanır. Tüm üyelerin uygunluk değerlerine bakıldığında, problem minimizasyon

problemiyse en küçük uygunluk değerine sahip üye, maksimizasyon problemiyse en büyük uygunluk değerine sahip üye, tüm üyeler içinde en iyi olanıdır. Yani problemin hedefine göre en iyi ve en kötü üyeler, uygunluk değerlerine bakılarak belirlenir.

4.1.4. Yerçekimi sabitinin güncellenmesi

Optimum çözüm aranırken, arama hızının kontrol edilmesi gerekmektedir. Her iterasyonda yerçekimi sabitini azaltarak bunu yapmak mümkündür. Dolayısıyla yerçekimi sabitinin hesaplanmasında aşağıdaki formül kullanılır.

$$G(t) = G_0 \times e^{\left(-\propto \frac{t}{T}\right)} \tag{E.4.2}$$

t anındaki yerçekimi sabiti ($G(t)$), $e^{\left(-\propto \frac{t}{T}\right)}$ ile yerçekimi sabitine verilen ilk değerin (G_0) çarpımıdır. Eşitlikte kullanılan T maksimum iterasyon sayısını, t o andaki iterasyon sayısını, α ise sabit bir değeri ifade eder.

4.1.5. Yay sabitinin güncellenmesi

Her iterasyonda yayların sertliği azaltılır. Böylelikle yayların sertliği kontrol edilebilir.

$$K(t) = k_0 \times e^{\left(-\propto \frac{t}{T}\right)} \tag{E.4.3}$$

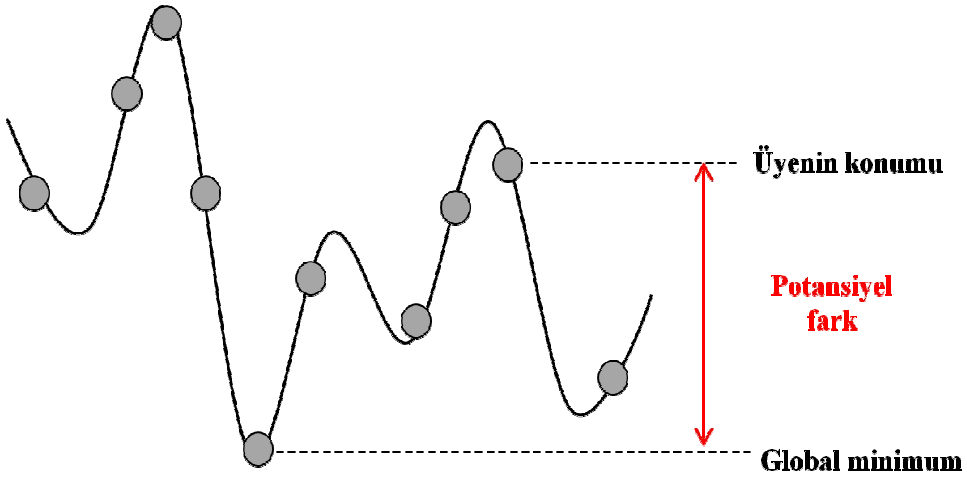
Yukarıda verilen eşitlikte t anındaki yay sabiti $K(t)$, yay sabitinin başlangıç değeri k_0, eşitliğin son bölümünde bulunan t, o andaki iterasyon sayısı, T maksimum iterasyon sayısı, α sabit bir sayıdır.

4.1.6. Kuvvetlerin hesaplanması

Global ve yerel minimumlara sahip bir minimizasyon probleminde üyeler arama uzayına rastgele dağıtıldıklarında uygunluk değerlerine göre global minimuma daha yakın ya da daha uzak mesafede bulunurlar.

Uygunluk değeri en küçük olan üye global minimuma en yakın olan üyedir. Şekil 4.1.'de üyeler arasındaki uzaklığın potansiyel farka neden olduğu görülmektedir. Bu potansiyel farktan dolayı da bir dış kuvvet oluşur. Yani dış kuvvet, üyelerin

uygunluk değerlerindeki farklılık nedeniyle, arama uzayındaki konumlarının oluşturduğu potansiyel farktan meydana gelir.



Şekil 4.1. Üyeler arasındaki potansiyel fark gösterimi.

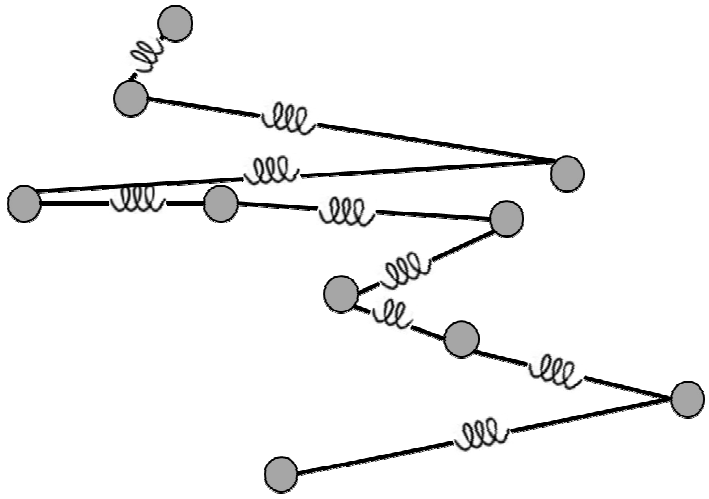
Üyeler arasında potansiyel farktan meydana gelen dış kuvvet aşağıdaki eşitlik ile hesaplanır:

$$F\_ext_{ij}^{d}(t) = G(t) \times \frac{fit_i(t) - worst(t)}{best(t) - worst(t)} \times \left(\frac{X_j^d(t) - X_i^d(t)}{\left| X_j^d(t) - X_i^d(t) \right| + \varepsilon} \right) \quad \text{(E.4.4)}$$

$$F\_ext_i^d(t) = \sum_{j \neq i}^{N} rand_j F\_ext_{ij}^{d}(t) \quad \text{(E.4.5)}$$

E.4.4.'de t anında j üyesinin i üyesi üzerinde oluşturduğu kuvvet hesaplanmıştır. $G(t)$ yerçekimi sabitini, $fit_i(t)$ t anında i. üyenin uygunluk değerini, $best(t)$ ve $worst(t)$ ise yine o andaki en iyi ve en kötü uygunluk değerlerine sahip çözümleri , $\left| X_j^d(t) - X_i^d(t) \right|$ i ve j üyelerinin d. boyuttaki birbirlerine olan uzaklıklarını, ε küçük sabit bir değeri ve $\left(X_j^d(t) - X_i^d(t) \right)$ ifadesi j. üye ile i. üyenin d. boyuttaki konum farklarını belirtir. Şekil 4.1. incelendiğinde global minimuma yakın olan üye ile uzak olan üye arasındaki potansiyel farkın yüksek olduğu görülür. Bu nedenle en yüksek potansiyele sahip olan üye global minimuma en yakın üye, en düşük potansiyele sahip olan üye ise global minimuma en uzak olan üyedir. Dolayısıyla potansiyeli yüksek olan üye diğer üyeleri kendine daha kuvvetli çeker.

Üyelerin birbirleri ile arasındaki potansiyel farktan doğan dış kuvvet hesaplandıktan sonra üyeler uygunluk değerlerine göre en iyiden en kötüye sıralanır. En iyi üyeden en kötü üyeye doğru sıralanmasının sebebi üyelerin en iyi üyeye doğru yay dolayısıyla bir çekim içerisine girmelerini sağlamaktır. Sıralanan üyeler arasına Şekil 4.2.'deki gibi yay bağlanarak kütle yay sistemi oluşturulur.



Şekil 4.2. En iyiden en kötüye doğru sıralanan üyeler arasına yay bağlanması.

Her üye yay ile bağlı olduğu üyelere yay dolayısıyla bir kuvvet uygular. Kütle yay sisteminde, yaylar ilk uzunluğunu korumak ister. Algoritmada kütle yay sisteminde kullanılan E.3.1. yerine aradaki potansiyel farkı sıfırlamak için aşağıdaki eşitlik kullanılmıştır.

$$\mathrm{F\_int}_{ij}^{d}(t) = K \times \frac{\mathrm{fit}_j(t) - \mathrm{fit}_i(t)}{\mathrm{best}(t) - \mathrm{worst}(t)} \times \left(\frac{X_j^d(t) - X_i^d(t)}{\left| X_j^d(t) - X_i^d(t) \right| + \varepsilon} \right) \tag{E.4.6}$$

$$\mathrm{F\_int}_{ji}^{d}(t) = -\mathrm{F\_int}_{ij}^{d}(t) \tag{E.4.7}$$

Yukarıdaki eşitliklerde j. üye, i. üyenin yay ile bağlı olduğu bir sonraki üyedir. E.4.6.'da iterasyon t'de d. boyutta j. üyenin i. üye üzerine uyguladığı kuvvet $\mathrm{F\_int}_{ij}^{d}(t)$, t anında i. üyenin uygunluk değeri $\mathrm{fit}_i(t)$, j. üyenin uygunluk değeri $\mathrm{fit}_j(t)$, t anındaki en iyi ve en kötü değerler $\mathrm{best}(t)$ ve $\mathrm{worst}(t)$, j. ve i. üyelerin d. boyuttaki konum farkı $X_j^d(t) - X_i^d(t)$ ve aralarındaki mesafe $X_j^d(t) - X_i^d(t)$ olarak gösterilmiştir. Eşitlikte kullanılan ε küçük bir sabit değerdir. Üyeler arasındaki yayın katsayısı ise K ile gösterilmiştir.

Kütle yay sisteminde yay ile bağlı olan kütleler yaydan dolayı bir kuvvete maruz kalırlar. Yay, kendini geren ya da sıkıştıran kuvvete eşit büyülükte ve ters yönde bir kuvvet uygular. Bu nedenle i. üye üzerine uygulanan kuvvet ile j. üye üzerine uygulanan kuvvet eşit büyüklükte ve zıt yönlüdür. E.4.7.'de kuvvetlerin eşit büyüklükte ve zıt yönde olduğu gösterilmiştir.

Yukarıdaki eşitliklere göre üyeler en iyi üyeye doğru bir çekim içerisine girer. Bu sayede arama uzayı içerisinde yerel arama işlemi de gerçekleşmiş olur. Son olarak bir üyenin diğer üyelere uyguladığı iç ve dış kuvvetler toplanarak toplam kuvvet hesaplanır.

$$F_i^d(t) = F_{int_i}^{\ d}(t) + F_{ext_i}^{\ d}(t) \tag{E.4.8}$$

E.4.8.'de $F_{int_i}^{\ d}(t)$ t anında d. boyuttaki i. üye üzerinde yay tarafından oluşan iç kuvveti, $F_{ext_i}^{\ d}(t)$ t anında d. boyuttaki i. üye üzerinde oluşan dış kuvveti ve $F_i^d(t)$ t anında d. boyutta i. üye üzerinde oluşan toplam kuvveti gösterir.

4.1.7. Üyelerin ivmelerinin hesaplanması

Newton'un hareket kanunlarından ikincisi olan ivme kanununa göre her bir üyenin ivmesinin hesaplandığı adımdır. Üye, kendisine etkiyen dış kuvvet ve iç kuvvetin bileşkesi yönünde ivme kazanır.

$$a_i^d(t) = \frac{F_i^d(t)}{M_i(t)} \tag{E.4.9}$$

Toplam kuvvet, üyenin o andaki kütlesine bölünerek ivme hesaplanır. E.4.9.'da $a_i^d(t)$ t anında i. üyenin d. boyuttaki ivmesini, $F_i^d(t)$ kendine etkiyen toplam kuvveti ve $M_i(t)$ kütlesini göstermektedir.

4.1.8. Hız ve konum güncellemesi

Üyenin $t+1$ anındaki hızı ve konumu explicit Euler integrasyonuna göre aşağıda verilen eşitlikler ile hesaplanır.

$$V_i^d(t+1) = rand_i \times V_i^d(t) + a_i^d(t) \tag{E.4.10}$$

$$X_i^d(t+1) = X_i^d(t) + V_i^d(t+1) \quad \text{(E.4.11)}$$

E.4.10.'da $i.$ üyenin $d.$ boyutta $t+1$ anındaki hızı $\left(V_i^d(t+1)\right)$, bir önceki hızı $\left(V_i^d(t)\right)$ ile hızında meydana gelen değişimin $\left(a_i^d(t)\right)$ toplamına eşittir. Bir önceki hız ile çarpılan $rand_i$ sıfır ile bir arasında seçilen rastgele bir sayıdır. E.4.11.'de ise bir sonraki konum$\left(X_i^d(t+1)\right)$, bir önceki konum$\left(X_i^d(t)\right)$ ile bir sonraki hızının toplamı ile hesaplanır.

4.1.9. Sonlandırma

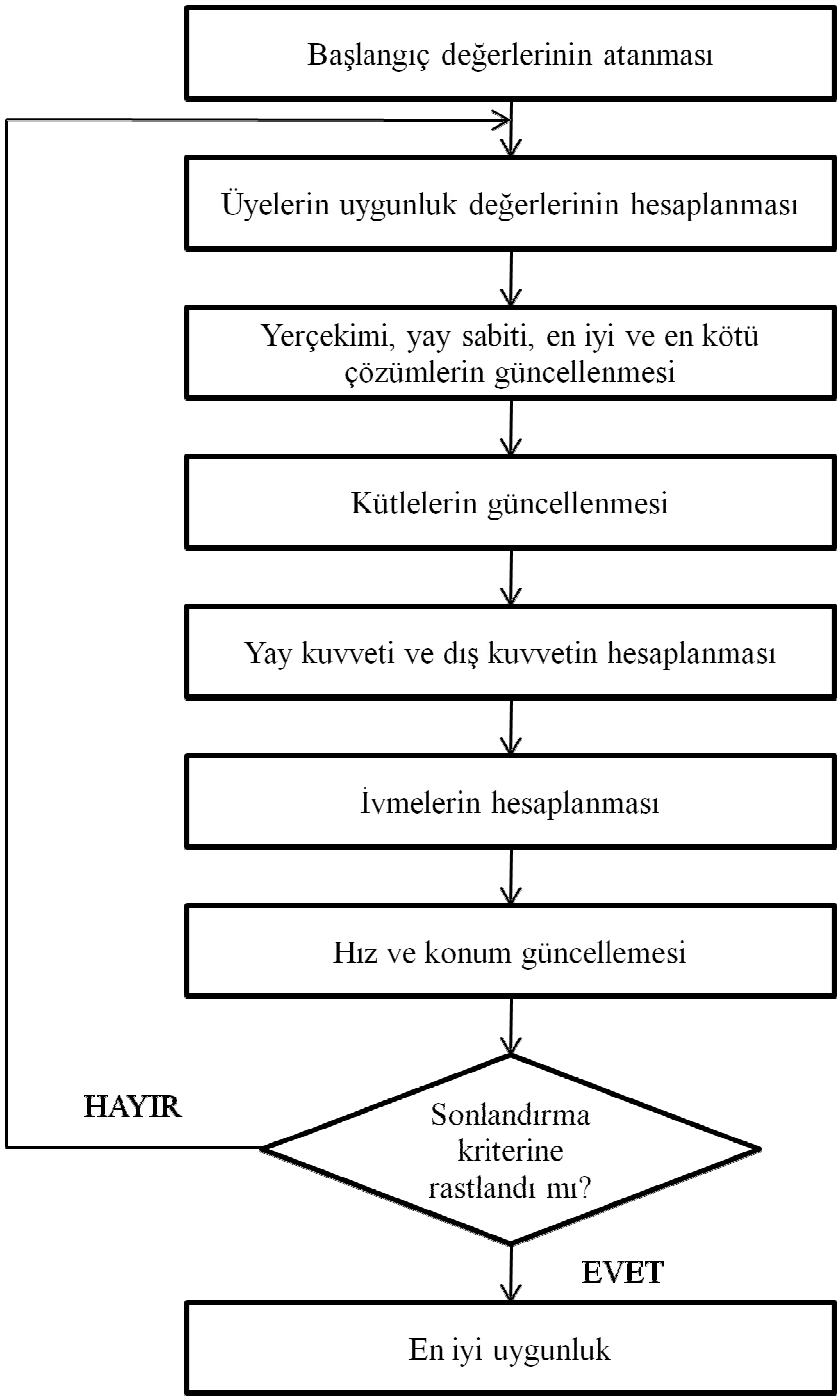
Algoritma herhangi bir sonlandırma kriterine uğradığında veya maksimum iterasyon sayısına ulaştığında durdurulur. Eğer şartlar sağlanmıyorsa 3. adıma geri dönülür ve şart sağlanana kadar devam ettirilir.

4.2. Algoritmanın Kaba Kodu

Başlangıç değerlerini belirle
FOR
N adet üyeyi rastgele konumlandır. $X_i = \left(x_i^1, \dots, x_i^d, \dots, x_i^n\right) \quad i = 1,2,\dots,N$
END
Üyelerin uygunluk değerlerini hesapla
Yer çekimi sabitini, yay sabitini, en iyi ve en kötü uygunluk değerlerini güncelle
DO FOR
Her üyenin dış kuvvetini hesapla
Üyeleri en iyiden en kötüye doğru sırala
Sıralanan üyeler arasına yay bağla
Her üyenin bağlı olduğu yaydan oluşan iç kuvvetini hesapla
İç ve dış kuvvetleri topla
Her üyenin ivmesini, hızını ve konumlarını güncelle
END
WHILE Durdurma koşulu sağlanan kadar devam et

4.3. Algoritmanın Akış Şeması

Şekil 4.3.'de MSS-GSA algoritmasının akış şeması verilmiştir.



Şekil 4.3. MSS-GSA algoritmasının akış şeması.

4.4. Uygulanan Test Fonksiyonları

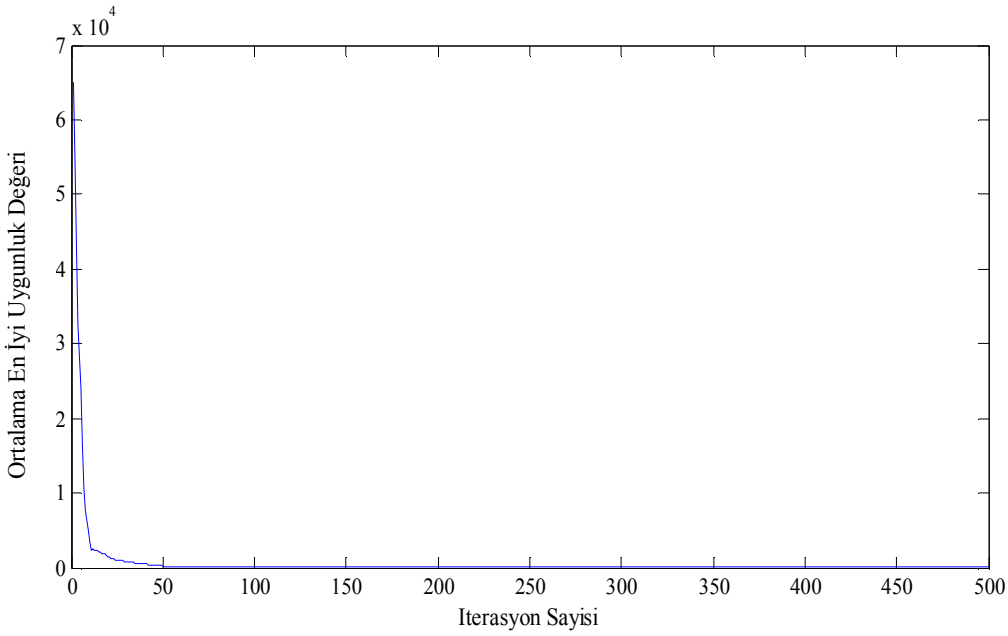
Önerdiğimiz algoritmanın etkinliğini gözlemlemek adına aşağıda verilen test fonksiyonları üzerinde MSS-GSA algoritması uygulanmıştır. Algoritmanın kodları ve fonksiyonlar MATLAB programlama dilinde yazılmış ve Intel(R) Core(TM) i5 CPU M

430 @ 2.27 GHz, 4GB RAM donanımlı, 32 bit işletim sistemine ait bir dizüstü bilgisayarda çalıştırılmıştır. Tüm fonksiyonlar için üyelerin kütlesi 10 alınmıştır.

4.4.1. Sphere fonksiyonu

E.2.17.'de fonksiyonun n boyutlu olduğu görülmektedir. Algoritma içerisinde n, 30 alınmıştır. Toplam üye adedi 25, maksimum iterasyon sayısı 500, değişken sınırları [-100,100], $\varepsilon = 10^{-12}$, yerçekimi sabitinin ilk değeri G_0= 50, yerçekimi sabitinin güncellenmesinde kullanılan α sabit değeri 10, yay sabitinin başlangıç değeri 50 olarak alınmış ve exponansiyel olarak azaltılmıştır.

Şekil 4.4.'de verilen grafiğe göre algoritma maksimum iterasyon sonunda en iyi uygunluk değerini 0,000000129020 olarak bulmuştur. Algoritmanın tamamlanması için geçen süre 9,767182 sn olarak tespit edilmiştir.

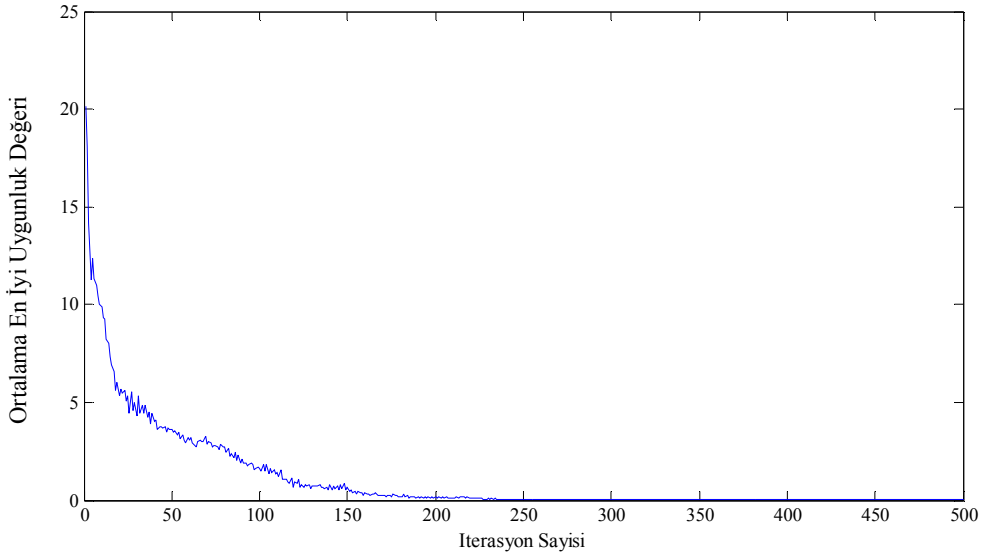


Şekil 4.4. MSS-GSA'nın 500 iterasyon boyunca Sphere fonksiyonunda ürettiği uygunluklar.

4.4.2. Ackley fonksiyonu

Algoritmada E.2.18.'de verilen Ackley fonksiyonunun boyutu 30, popülasyon büyüklüğü, maksimum iterasyon sayısı, ε, α, G_0 ve yay sabitinin başlangıç değeri Sphere fonksiyonunda kullanılan değerlere eşit alınmıştır.

Şekil 4.5.'de 500 iterasyon sonunda algoritma, minimum uygunluk değerini 0.000222310561 olarak 13.725497 sn'de hesaplamıştır.

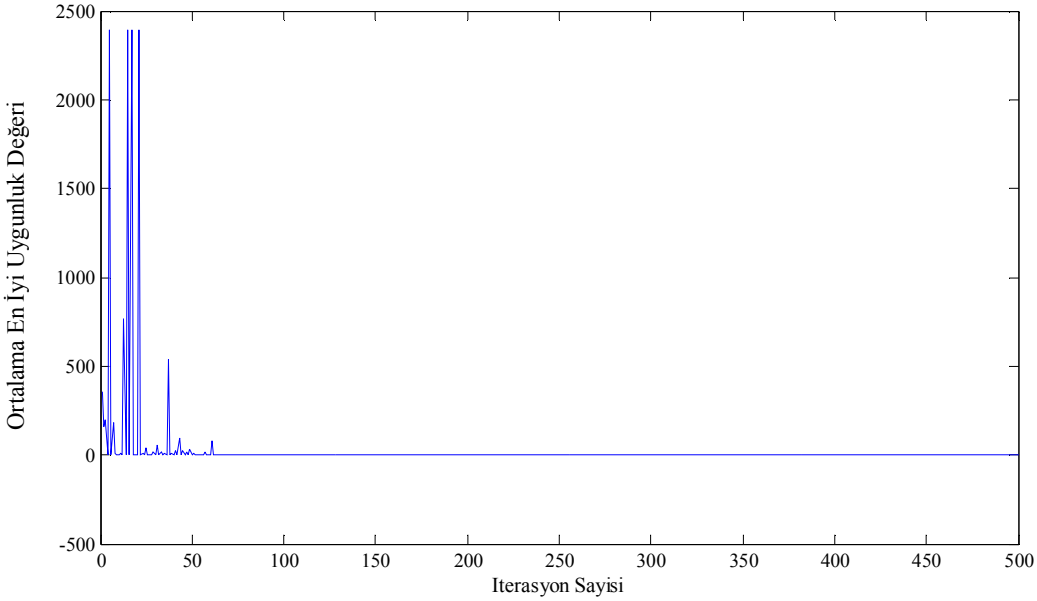


Şekil 4.5. MSS-GSA'nın 500 iterasyon boyunca Ackley fonksiyonunda ürettiği uygunluklar.

4.4.3. Humpcamel fonksiyonu

2 boyuta sahip olan Humpcamel fonksiyonu E.2.19.'da gösterilmiştir. MSS-GSA algoritması ile global minimuma erişilirken, toplam üye sayısı 25, maksimum iterasyon sayısı 500, değişken sınırları [-5,5], $\varepsilon = 10^{-12}$, yerçekimi sabiti hesabında α = 10, G_0= 50, yay sabitinin başlangıç değeri 50 olarak alınmıştır.

Şekil 4.6.'da verilen grafikte 500 iterasyon sonunda global minimum -1.031628266185 olarak bulunmuştur. Algoritmanın tamamlanması için geçen süre 15.540520 sn olarak hesaplanmıştır.



Şekil 4.6. MSS-GSA'nın 500 iterasyon boyunca Humpcamel fonksiyonunda ürettiği uygunluklar.

5. ÜYE – UYDU ALGORİTMASI

Arama uzayına rastgele konumlandırılan her bir obje, önerilen algoritmada üye olarak isimlendirilir. Üyeler etrafına önceden belirlenen bölge içerisine belli miktarda uydular atanır. Üye ve uydularının uygunluk fonksiyonuna göre uygunluk değerleri hesaplanır. İçlerinden en iyi uygunluk değerine sahip olanı üyelik sıfatını alır. Eğer önceden var olan üye daha iyi bir uygunluk değerine sahipse algoritma olduğu gibi devam ettirilir.

Üye-Uydu algoritmasında her üyeye atanan uydular da optimum noktayı arayışa geçer. Uyduların uygunluk değerlerinin hesaplanması dışında algoritma içerisinde diğer hesapsal işlemlere dâhil edilmemesi algoritmanın hesapsal yükünü azaltır. Bu nedenle algoritmanın diğer algoritmalara göre daha etkili olması beklenmektedir.

5.1. Algoritma Adımları

Algoritma adımları aşağıdaki şekilde verilebilir.

5.1.1. İlk değerlerin atanması

Algoritmada kullanılan sabitlere değer atamaları yapılır. Algoritma süresince her iterasyonda değişen değişkenlere ilk değerleri verilir.

5.1.2. Arama uzayının tasarlanması

Arama uzayının büyüklüğü problemin sınırları göz önüne alınarak belirlenir. Üyeler arama uzayına rastgele dağıtılır.

$$X_i = \left(x_i^1, \dots, x_i^d, \dots, x_i^n\right) \qquad i = 1,2, \dots, N \qquad \text{(E.5.1)}$$

x_i^d, i. üyenin d. boyuttaki konumunu belirtir.

5.1.3. Uyduların belirlenmesi

Üyelerin optimum noktayı daha iyi ve daha hızlı bulabilmesi amacıyla her üye için sabit sayıda uydu tanımlanır. Şekil 5.1.'de siyah çerçeveli ve çapı diğer dairelere göre büyük olan daireler üyeleri ifade etmektedir. Her üye kendisine ait, aynı renkte

tanımlanmış uydulara sahiptir. Uydular, üyelere belli bir çap içerisinde rastgele yerleştirilir.

Şekil 5.1. Üye-uydu gösterimi.

N adet üyeye ve her bir üyenin M adet uyduya sahip olduğu bir arama uzayı düşünüldüğünde i. üyenin s. uydusunun konumu aşağıdaki gibi tanımlanır:

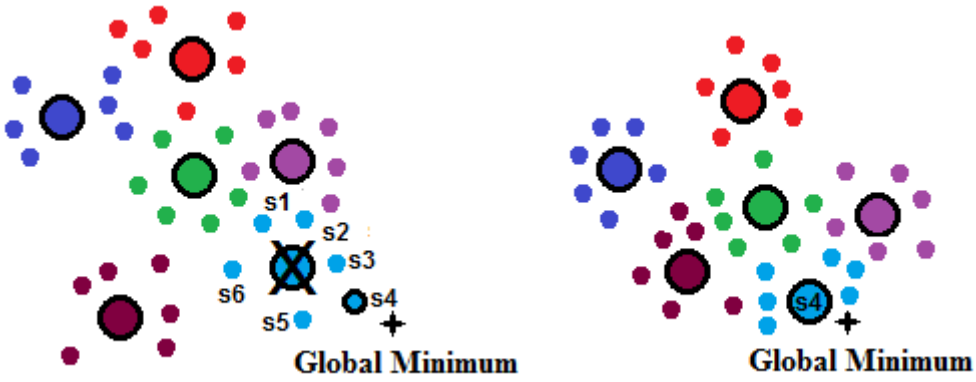
$$X_{is} = \left(x_{is}^{1}, \dots, x_{is}^{d}, \dots, x_{is}^{n}\right) \qquad i = 1,2,\dots,N \qquad s = 1,2,\dots,M \qquad \text{(E.5.2)}$$

Yukarıdaki tanımlamada x_{is}^{d} i. üyenin s. uydusunun d. boyuttaki konumunu göstermektedir.

5.1.4. Uygunluk değerlerinin hesaplanması

Arama uzayındaki her üye ile bu üyelerin uyduları uygunluk fonksiyonuna gönderilir. Elde edilen uygunluk değerleri üye ile uydular arasında karşılaştırılır. Fonksiyona gönderilen üye, uydularından daha iyi bir değer vermişse üyeliğini devam ettirir ve konumunu korur. Uydunun uygunluk değeri, sahip olduğu üyenin uygunluk değerinden daha iyiyse, belirlenen uydu, sistemin yeni üyesidir ve algoritma adımları yeni üyenin konumuna göre işlemeye devam eder.

Şekil 5.1.'de farklı renklere sahip üyeler ve kendileriyle aynı renkte olan uyduları gösterilmiştir. Tüm üyeler ve uyduları uygunluk fonksiyonuna gönderildiğinde açık mavi renkli üyenin 4 numaralı uydusu (s4) global minimuma daha yakın olduğu için daha iyi bir değer çıkarmıştır. Bu nedenle şekil 5.2.'de görüldüğü üzere s4 uydusu artık yeni üye olmuştur. Bir sonraki işlemler s4 üyesi üzerinden gerçekleştirilir.



Şekil 5.2. Minimuma yakın olan uydunun üye olması.

Elde edilen yeni üyeler ile birlikte tüm üyeler içinde en iyi ve en kötü üye seçilir. Minimizasyon problemlerinde best(t) ve worst(t) E.2.4. ile E.2.5. kullanılarak belirlenir. Maksimizasyon problemlerinde ise E.2.6. ve E.2.7. kullanılarak best(t) ve worst(t) değerleri seçilir.

5.1.5. Yerçekimi sabitinin güncellenmesi

Algoritma boyunca yerçekimi sabiti azaltılarak arama kontrol edilir. E.2.3.'de yerçekimi sabitinin başlangıç değeri her iterasyonda exponansiyel olarak azaltılmıştır.

5.1.6. Kütlelerin hesaplanması

Arama uzayında bulunan her bir üyenin kütle değerleri E.2.9.'a göre hesaplanır. E.2.10.'da ise her üyenin kütlesi toplam kütleye bölünerek normalizasyon işlemi gerçekleştirilir.

5.1.7. Kuvvetlerin hesaplanması

Bir üyeye diğer tüm üyelerin etkidiği toplam kuvvettin hesaplandığı adımdır. E.2.12. ile j. üye tarafından i. üyeye etkiyen kuvvet, Newton'un evrensel çekim kanununa göre hesaplanmıştır. Daha sonra E.2.13. ile tüm üyelerin i. üyeye yaptığı toplam kuvvet hesaplanır.

5.1.8. İvmelerin hesaplanması

Newton'un 2. hareket kanununa göre bir cismin ivmesi, cisme uygulanan kuvvet ile kütlesinin bölümüne eşittir. Algoritmada E.2.14. ile ivme hesaplanırken Newton'un hareket kanunu esas alınmıştır.

5.1.9. Hız ve konum güncellemesi

Üyenin $t+1$ anındaki hızı ve konumu explicit Euler integrasyonuna göre sırasıyla E.2.15. ve E.2.16. kullanılarak hesaplanır.

5.1.10. Sonlandırma

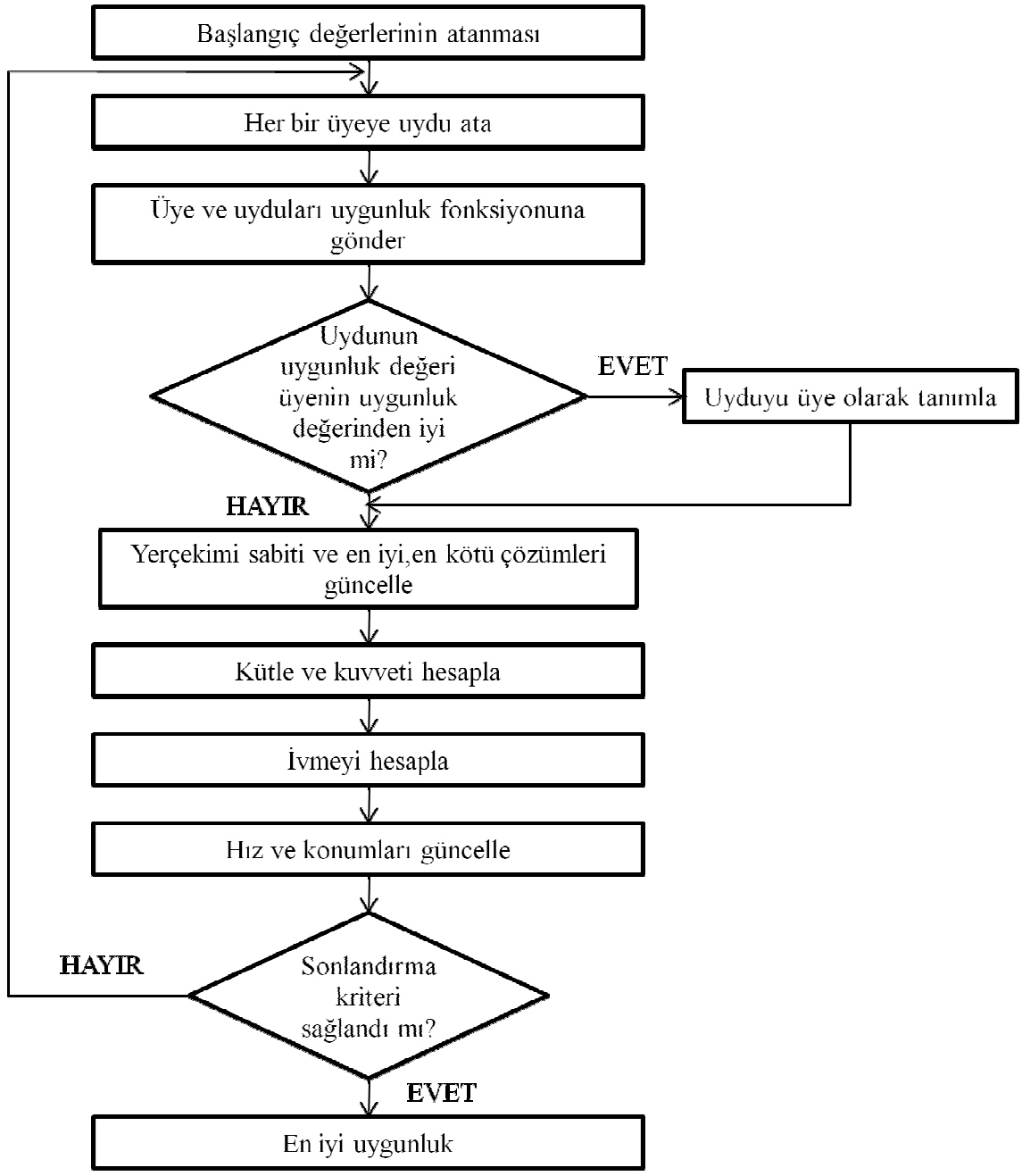
Maksimum iterasyon sayısı tamamlandığında ya da bir sonlandırma kriterine rastlandığında algoritma durdurulur. Aksi durumda 3. adıma dönülerek, şartlar sağlanana kadar algoritma devam ettirilir.

5.2. Algoritmanın Kaba Kodu

Başlangıç değerlerini ata
FOR
N adet üyeyi arama uzayı içerisine rastgele konumlandır.
$X_i = \left(x_i^1, \dots, x_i^d, \dots, x_i^n\right) \quad i = 1,2, \dots, N$
END
DO FOR
N adet üye için belli bir bölge içerisinde M sayıda uyduyu rastgele konumlandır.
$X_{is} = \left(x_{is}^1, \dots, x_{is}^d, \dots, x_{is}^n\right) \quad i = 1,2, \dots, N \quad s = 1,2, \dots, M$
Tüm üye ve uydularını uygunluk fonksiyonuna gönder, her üyenin uygunluk değeri ile uydularının uygunluk değerlerini karşılaştır, eğer uydularından birinin uygunluk değeri üyesinden iyi ise, uyduyu üyeye eşitle.
Yerçekimi sabitini ve en iyi, en kötü çözümleri güncelle
Üyelerin kütlelerini hesapla
Her bir üyenin üzerine etkiyen kuvveti hesapla
İvmelerini hesapla
Hız ve konumlarını güncelle
END
WHILE Sonlandırma kriteri sağlanana kadar devam et

5.3. Algoritmanın Akış Şeması

Şekil 5.3.'de Üye-Uydu algoritmasının akış şeması verilmektedir.



Şekil 5.3. Üye-Uydu algoritmasının akış şeması.

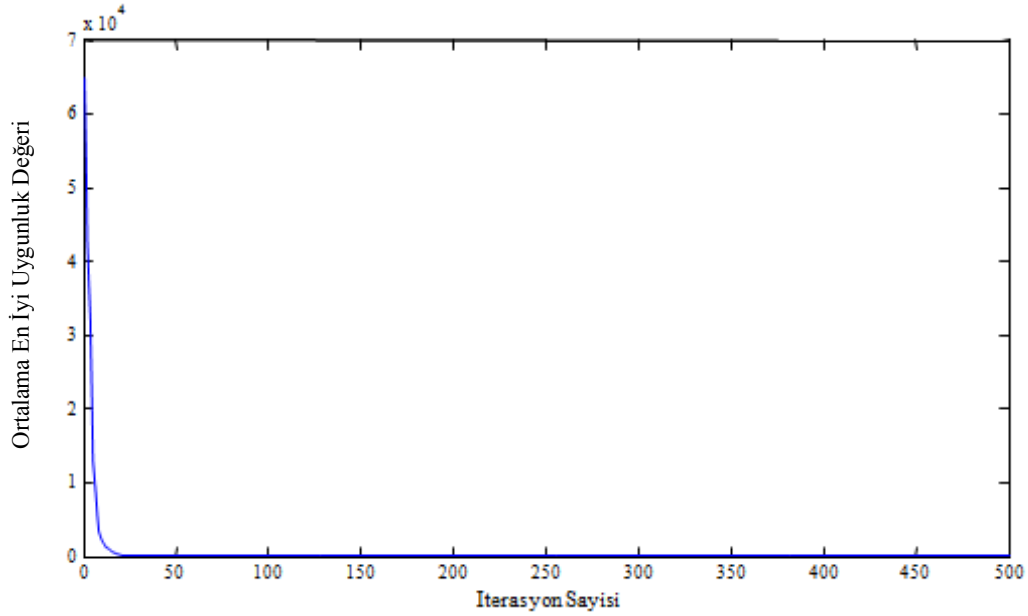
5.4. Uygulanan Test Fonksiyonları

Üye-Uydu algoritmasının başarısını test etmek amacıyla 3 adet test fonksiyonu kullanılmıştır. Bunlar; Sphere, Ackley ve Humpcamel fonksiyonlarıdır. Üye-Uydu algoritmasının kodları ve fonksiyonlar MATLAB programlama dilinde yazılmış ve Intel(R) Core(TM) i5 CPU M 430 @ 2.27 GHz, 4GB RAM donanımlı, 32 bit işletim sistemine ait bir dizüstü bilgisayarda çalıştırılmıştır.

Algoritmada, her test fonksiyonu için toplam üye sayısı 25, her üyenin uydu sayısı 6, uyduların yerleşeceği alan en yakın iki uydunun arasındaki mesafenin yarısı, ε = 10^{-12}, α = 10 ve G_0= 50 olarak alınmıştır. Başlangıç değerleri her kullanıcıya göre değiştirilebilir.

5.4.1. Sphere fonksiyonu

E.2.17.'de verilen Sphere fonksiyonu için, algoritmada fonksiyon boyutu 30, maksimum iterasyon sayısı 500, değişken sınırları [-100,100] olarak alınmıştır. Şekil 5.4.'de ise algoritma süresince elde edilen uygunluk değerleri gösterilmiştir.



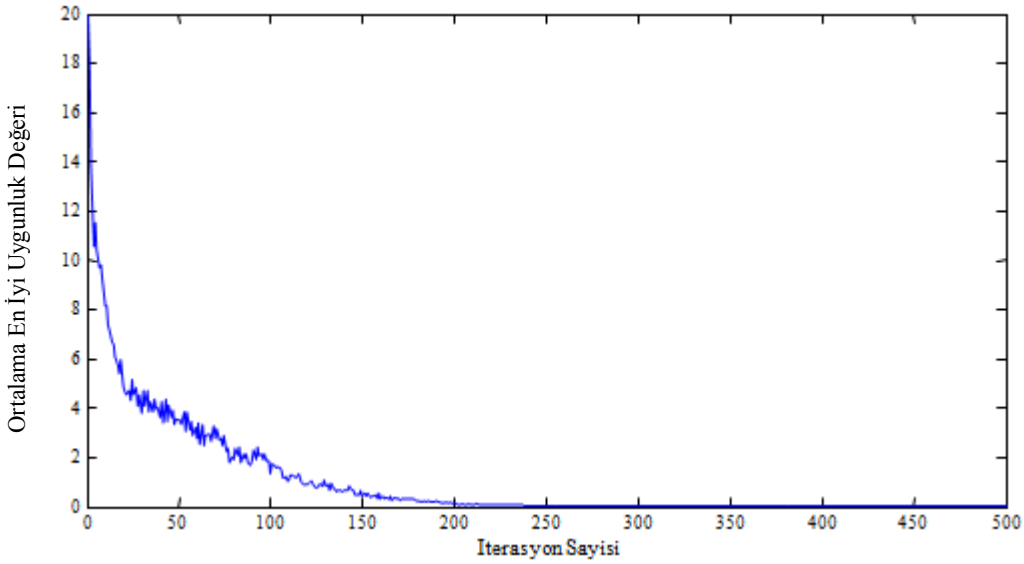
Şekil 5.4. Üye-Uydu algoritmasının 500 iterasyon boyunca Sphere fonksiyonunda ürettiği uygunluklar.

Şekil 5.4. incelendiğinde 500 iterasyon sonunda, en iyi uygunluk değeri 0.000000086662 olarak bulunmuştur. Algoritmanın tamamlanabilmesi için geçen süre 14.027651 olarak hesaplanmıştır.

5.4.2. Ackley fonksiyonu

E.2.18.'de verilen Ackley fonksiyonu için, algoritmada fonksiyonun boyutu 30 olarak belirlenmiş olup, maksimum iterasyon sayısı 500 ve değişken sınırları [-32,32] olarak alınmıştır.

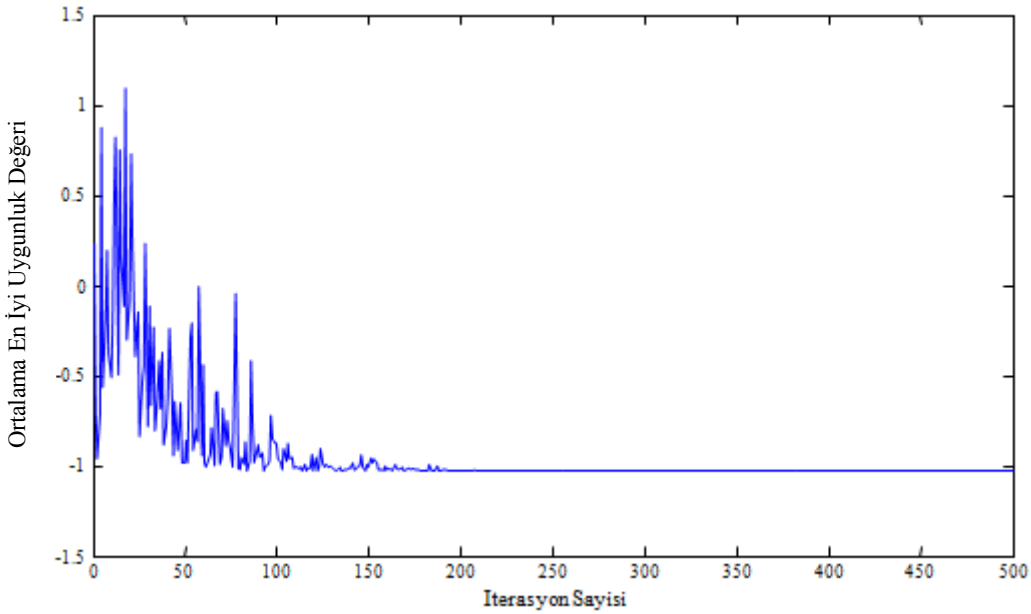
Şekil 5.5.'de algoritma boyunca elde edilen çözümler görülmektedir. Maksimum iterasyon sonucunda algoritma en iyi uygunluk değerini 0.000237210267 olarak bulmuştur. Hesaplama için geçen süre 14.802977 sn'dir.



Şekil 5.5. Üye-Uydu algoritmasının 500 iterasyon boyunca Ackley fonksiyonunda ürettiği uygunluklar.

5.4.3. Humpcamel fonksiyonu

E.2.19.'da verilen Humpcamel fonksiyonu için, Üye - Uydu algoritmasında maksimum iterasyon sayısı 500 ve değişken sınırları [-5,5] olarak alınmıştır. Şekil 5.6.'da maksimum iterasyon sayısı sonunda en iyi uygunluk değeri -1.031628447203 olarak 14.403749 sn'de bulunmuştur.



Şekil 5.6. Üye-Uydu algoritmasının 500 iterasyon boyunca Humpcamel fonksiyonunda ürettiği uygunluklar.

6. SONUÇLAR

Bu tez çalışmasında optimizasyon problemlerinin çözümü için Rashedi tarafından önerilen yerçekimine dayalı arama algoritmasından esinlenerek iki adet yeni optimizasyon yöntemi geliştirilmiştir. Metasezgisel algoritmalardan olan YAA'da arama uzayında arama yapılırken Newton'un evrensel çekim kanunu göz önüne alınmıştır. Kütleler minimuma yaklaştıkça en iyi objenin kütlesi artırılmış ve büyük kütleli objeler diğer kütleleri kendine daha kuvvetli çekmiştir. Böylelikle tüm kütleler en iyi kütleye doğru hızla ilerlemektedir. Algoritma tamamlandığında en büyük kütleye sahip obje en iyi çözümü vermektedir.

Geliştirilen algoritmalardan ilki MSS-GSA algoritmasıdır. Algoritmada arama uzayındaki kütleler iki farklı kuvvete maruz kalmaktadır. Bunlardan ilki arama uzayındaki konumlarından dolayı birbirlerine oluşturdukları potansiyel farktan oluşan kuvvettir. En yüksek potansiyele sahip olan kütle, minimuma en yakın olan kütledir. Dolayısıyla diğer kütleler ile arasında oluşturduğu kuvvet daha büyüktür. Sistem iki kütle arasındaki potansiyeli sıfıra indirgemek ister. Böylelikle minimuma yakın kütle, diğerlerini kendine daha kuvvetli çekerek tüm kütlelerin minimuma yakınsanmasını sağlamıştır. Objelerin maruz kaldığı diğer kuvvet ise kütleler arasına bağlanan yaydan dolayı oluşan kuvvettir. Objeler uygunluk değerlerine bakılarak en iyiden en kötüye doğru sıralanır. Sıralanan objeler arasına yay bağlanarak birbirlerinden bağımsız hareket etmeleri engellenir. Oluşan kütle yay sistemi ile kütleler birbirlerine kuvvet uygulayarak en iyi çözüme doğru çekilirler. Böylelikle MSS-GSA algoritmasında hem global hem yerel arama gerçekleştirilmiş olunur.

Geliştirilen diğer bir algoritma da üye-uydu algoritmasıdır. Algoritma YAA'ya benzer çalışır. Üye-uydu algoritmasında optimumu arayan objeler üye olarak isimlendirilir. Algoritmanın YAA ile ayrıldığı nokta üyelerin sahip olduğu uydulardır. Üyelerle birlikte uydular da optimum noktayı ararlar. Arama işlemi gerçekleştirilirken minimuma en yakın olan üye veya uyduya bakılır. Üye daha yakın ise algoritma YAA gibi devam eder. Fakat uydu yakın ise yeni üye, minimuma yakın olan uydu olur ve yeni üye üzerinden algoritma devam ettirilir. Böylelikle optimum nokta aranırken daha iyi bir arama gerçekleştirilir.

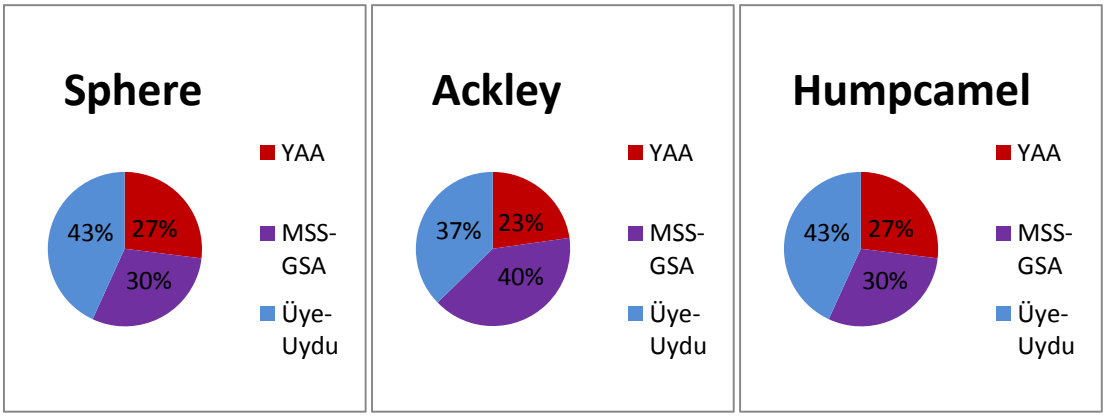
MSS-GSA algoritması ve üye-uydu algoritmasının etkinliğini test edebilmek amacıyla bazı test fonksiyonları kullanılmıştır. Elde edilen sonuçlar Çizelge 6.1.'de gösterilmiştir.

Çizelge 6.1. Algoritmaların test fonksiyonları üzerinde verdiği sonuçlar.

Fonksiyon		**YAA**	**MSS-GSA Algoritması**	**Üye-Uydu Algoritması**
Sphere	**En iyi uygunluk**	0.00012169299	0.000000129020	0.000000086662
	Hata	0.00012169299	0.000000129020	0.000000086662
	Hesaplama süresi	8.778853	9.767182	14.027651
Ackley	**En iyi uygunluk**	0.00427427807	0.000222310561	0.000237210267
	Hata	0.00427427807	0.000222310561	0.000237210267
	Hesaplama süresi	9.347089	13.725497	14.802977
Humpcamel	**En iyi uygunluk**	-1.03162793160	-1.03162826618	-1.03162844720
	Hata	0.00002793160	0.00002826618	0.00002844720
	Hesaplama süresi	8.761347	15.540520	14.403749

Çizelge 6.1.'e göre geliştirilen algoritmaların YAA'ya göre daha iyi uygunluk değerleri elde ettiği görülmektedir. Sphere fonksiyonunda, 0.000000086662 ile Üye-Uydu algoritması en iyi uygunluğu üretmiştir. En kötü uygunluğa ise YAA ulaşmıştır. Ackley fonksiyonunda 0.000222310561 ile en iyi uygunluğu MSS-GSA algoritması, en kötü uygunluğu ise 0.00427427807 ile YAA'nın verdiği görülmektedir. Uygulanan son fonksiyon olan Humpcamel fonksiyonunda tüm algoritmalar birbirlerine çok yakın değerde uygunluklar üretmişlerdir.

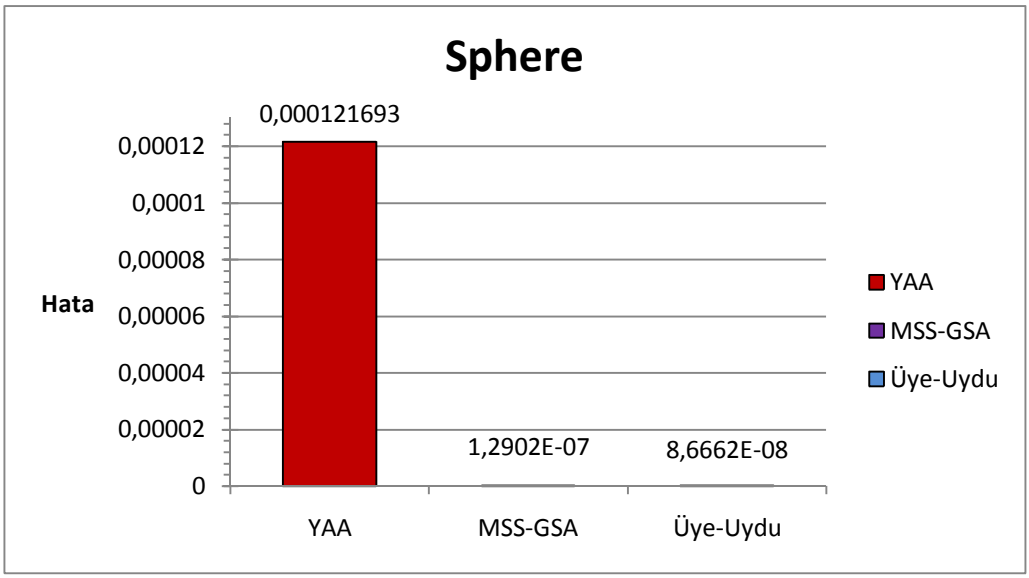
Şekil 6.1.'de algoritmaların hesaplama süreleri uygulanan fonksiyonlara göre karşılaştırıldığında en kısa zamanda çözüme ulaşan algoritmanın YAA olduğu görülmektedir. Geliştirilen algoritmaların YAA'ya göre daha uzun sürede hesaplama yaptığı sonucuna varılmıştır.



(a) (b) (c)

Şekil 6.1. a) Algoritmaların Sphere fonksiyonunda en iyi uygunluğa ulaşma süreleri b) Algoritmaların Ackley fonksiyonunda en iyi uygunluğa ulaşma süreleri c) Algoritmaların Humpcamel fonksiyonunda en iyi uygunluğa ulaşma süreleri.

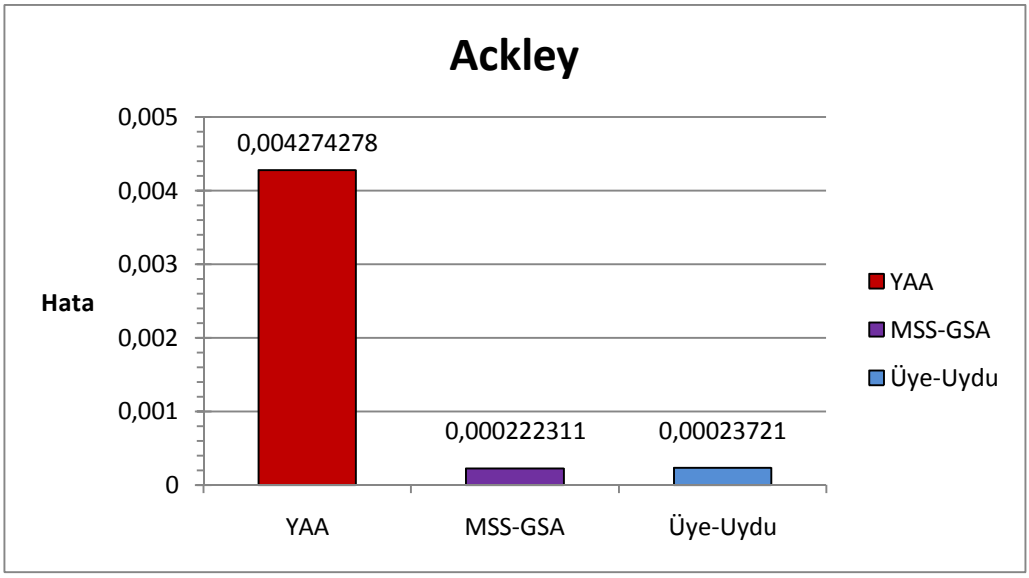
Test fonksiyonlarının optimum noktaları ile kullanılan algoritmalar sonucunda elde edilen değerler arasındaki fark, algoritmaların hata miktarlarını vermektedir. Şekil 6.2.'de YAA, MSS-GSA, Üye-Uydu algoritmasının Sphere fonksiyonu üzerinde uygulanması sonucunda elde edilen hata miktarları verilmektedir. Üye-Uydu algoritması en az hata ile çalışmıştır. MSS-GSA ile Üye-Uydu algoritması birbirlerine yakın değerler vermiştir. Fakat YAA algoritması geliştirilen algoritmalara göre daha çok hata yaparak, en kötü sonucu vermiştir.



Şekil 6.2. Algoritmaların Sphere fonksiyonundaki hata miktarları.

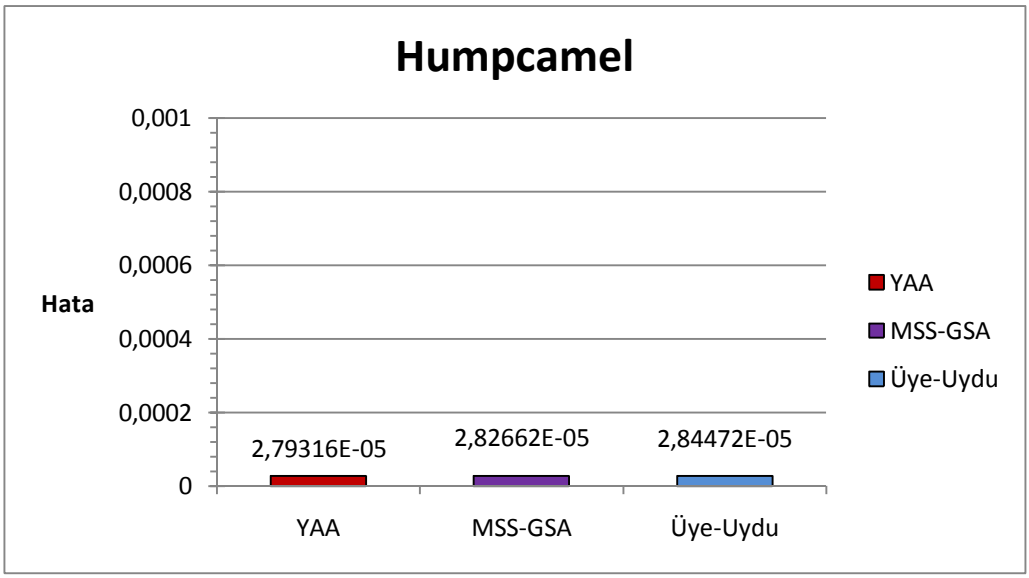
Şekil 6.3.'de ise belirlenen optimizasyon algoritmalarının Ackley fonksiyonu üzerinde uygulanmaları sonucu elde edilen hata miktarları verilmiştir. Elde edilen

sonuçlara göre MSS-GSA algoritması en az hata ile çalışmış olup Üye-Uydu algoritması ile benzer sonuçlar üretmiştir. Ackley fonksiyonunda da en fazla hata ile değer üreten algoritmanın YAA olduğu görülmektedir.



Şekil 6.3. Algoritmaların Ackley fonksiyonundaki hata miktarları.

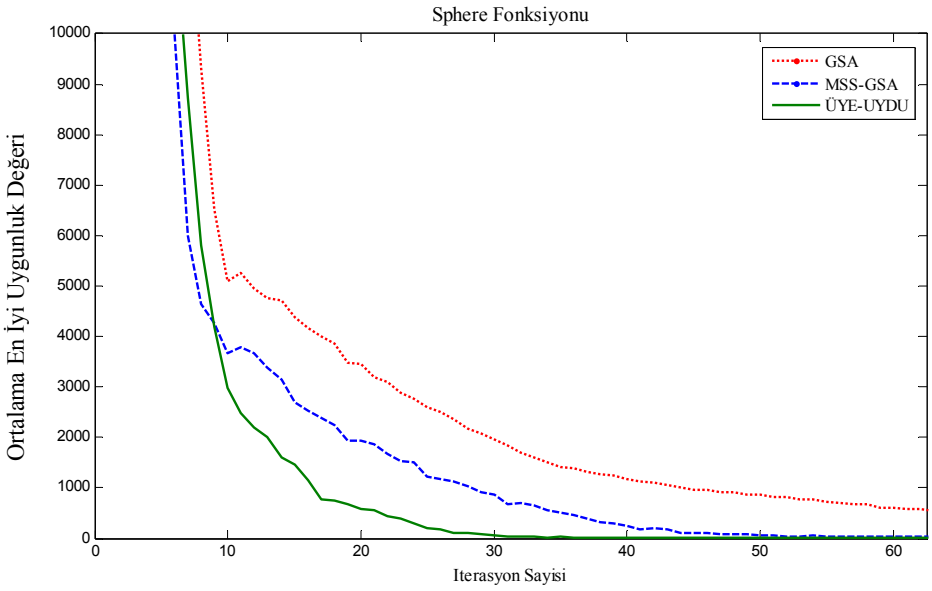
Humpcamel fonksiyonunda ise Şekil 6.4.'de de görüldüğü gibi YAA, MSS-GSA ve Üye-Uydu algoritmaları global minimuma sıfıra yakın hatalarla ulaşmışlardır. YAA, yeni geliştirilen algoritmalara göre yaklaşık 0.000001 kadar daha iyi sonuçlar elde etmiştir. Dolayısıyla diğer algoritmalara göre hata miktarı daha azdır.



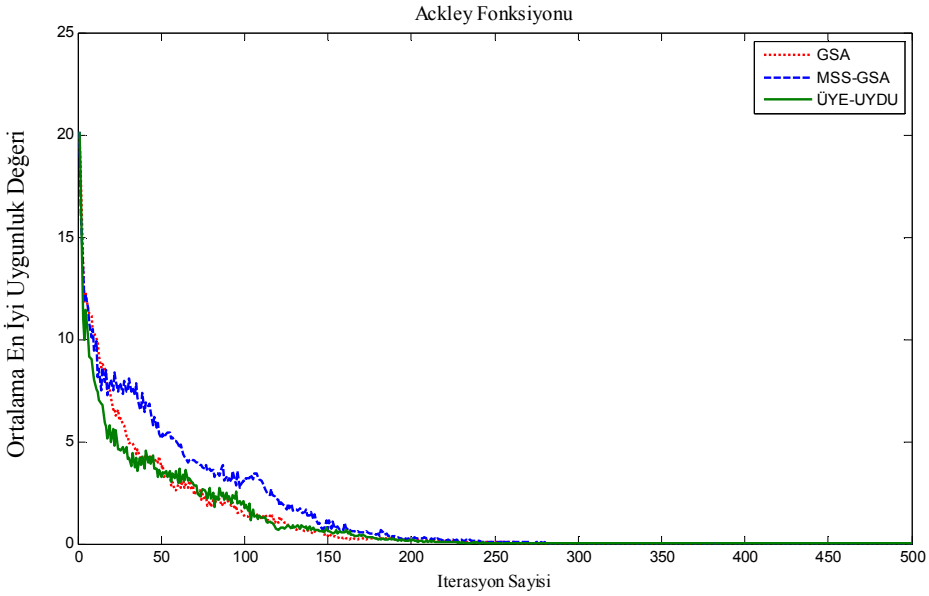
Şekil 6.4. Algoritmaların Humpcamel fonksiyonundaki hata miktarları.

Her bir algoritmanın başarım kıyaslaması her üç fonksiyon için Şekil 6.5., Şekil 6.6. ve Şekil 6.7.'de verilmiştir. Şekiller incelendiğinde Üye-Uydu algoritmasının diğer

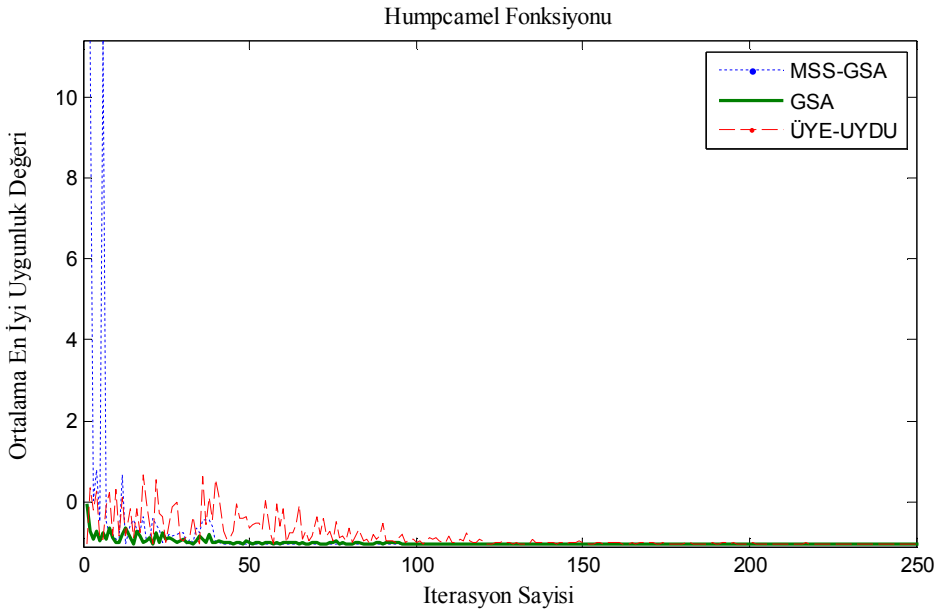
algoritmalara göre daha iyi uygunluklar ürettiği gözlemlenmiştir. YAA ise diğer algoritmalara göre daha kötü uygunluklar elde etmiştir.



Şekil 6.5. Algoritmaların Sphere fonksiyonundaki başarım kıyaslamaları.



Şekil 6.6. Algoritmaların Ackley fonksiyonundaki başarım kıyaslamaları.



Şekil 6.7. Algoritmaların Humpcamel fonksiyonundaki başarım kıyaslamaları.

Yapılan tez çalışmasında geliştirilen optimizasyon yöntemlerinin hesaplama sürelerinin uzun olmasına rağmen son zamanlarda popüler bir optimizasyon yöntemi olan YAA'ya göre problemlere daha iyi çözümler ürettiği gözlemlenmiştir. Bundan sonra yapılacak çalışmalarda sunulan algoritmaların daha kısa sürede çözüme ulaşmalarını sağlamak amaçlanacaktır.

KAYNAKLAR

Alaykıran, K. Ve Engin, O., "Karınca Kolonileri Metasezgiseli ve Gezgin Satıcı Problemleri Üzerinde Bir Uygulaması", ***Gazi Üniv. Müh. Mim. Fak. Der***., 20(1): 69-76 (2005).

Beasley, D., Bull, D.R. ve Martin, R.R., "An Overview of Genetic Algorithm:Part1-Fundamentals", ***Univ. of Michigan***, 15(2): 58-69 (1993).

Blum, C. ve Roli, A., "Metaheuristics in Combinatorial Optimization: Overview and Conceptual Comparison", ***ACM Computing Surveys***, 35(3): 268-308 (2003).

Ceylan, O., Özdemir, A. ve Dağ, H., "Gravitational Search Algorithm for Post-Outage Bus Voltage Magnitude Calculations", ***UPEC2010***, United Kingdom, (2010).

Chatterjee, A. ve Mahanti, G. K., "Comparative Performance Of Gravitational Search Algorithm And Modified Particle Swarm Optimization Algorithm For Synthesis Of Thinned Scanned Concentric Ring Array Antenna", ***Progress In Electromagnetics Research B***, 25: 331-348 (2010).

Chatterjee, A., Mahanti G. K. ve Mahapatra P. R. S., "Generation of Phase-only Pencil-beam Pair from Concentric Ring Array Antenna Using Gravitational Search Algorithm", ***ICCSP'11***, 384-388, (2011).

Deb, K., "A population-based algorithm-generator for real-parameter optimization", ***Soft Computing - a Fusion of Foundations, Methodologies and Applications***, 9(4): 236-253 (2005).

Dorigo, M. ve Di Caro, G., "Ant Colony Optimization: A New Meta-Heuristic", ***Proceedings of the Congress on Evolutionary Computation***, Washington, 2: 1470-1477 (1999).

Dorigo, M., Di Caro, G. ve Gambardella, L. M., "Ant Algorithms for Discrete Optimization", ***Artificial Life***, 5: 137-172 1999.

Duysak, A., Zhang, J.J. ve Ilankovan, V., "Efficient modelling and simulation of soft tissue deformation using mass-spring systems", ***International Congress Series***, 1256: 337-342, (2003).

Duysak, A., "Yüz Dokularındaki Deformasyonların Kütle-Yay Metodu Kullanılarak Gerçekçi Simülasyonu", ***SİU'05***, Kayseri, (2005).

Duysak, A**.,** Zhang, J. J, "Fast Simulation of Facial Tissue Deformations Using Mass-Spring Chain Algorithm", ***Theory and Practice of Computer Graphics, the 23rd Conference organized by the UK chapter of the Euro Graphics Association****, 139-145*, (2005).

Eberhart, R. C. ve Shi, Y., "Particle swarm optimization: developments, applications and resources", ***Proc. Congress On Evolutionary Computation 2001 IEEE Service Center***, 1: 81-86 (2001).

KAYNAKLAR (Devam Ediyor)

Erdoğmuş, P., "Particle Swarm Optimization Performance on Special Linear Programming Problems", ***Scientific Research and Essays***, 5(12): 1506-1518 (2010).

Erol, V., **"**Araç rotalama problemleri için popülasyon ve komşuluk tabanlı metasezgisel bir algoritmanın tasarımı ve uygulaması", Yüksek Lisans Tezi, ***Yıldız Teknik Üniversitesi Fen Bilimleri Enstitüsü***, İstanbul, 2006.

Goldberg, D.E., " Genetic Algorithms in Search Optimization and Machine Learning", ***Addison Wesley Publishing Company***, USA, 1989.

Karaboğa, D., "Yapay Zeka Optimizasyon Algoritmaları", ***Atlas Yayın Dağıtım***, İstanbul, 2004.

Kaya, S. ve Engin, O., "Sabit iş çizelgeleme problemleri: Literatür araştırması ve meta sezgisel yöntemler ile çözüm önerisi", ***İtüdergisi/d Mühendislik***, 8(1): 37-47 (2009).

Kennedy, J. ve Eberhart, R.C., "Particle Swarm Optimization," ***Proceedings Of The IEEE International Conference On Neural Networks***, Australia, 1942-1948 (1995).

Keskintürk, T., "Diferansiyel Gelişim Algoritması", ***İstanbul Ticaret Üniversitesi Fen Bilimleri Dergisi***, 5(9): 85-99 (2006).

Keskintürk, T. ve Şahin, S., "Doğrusal Olmayan Regresyon Analizinde Gerçek Değer Kodlamalı Genetik Algoritma", ***İstanbul Ticaret Üniversitesi Sosyal Bilimler Dergisi,*** 8(15): 167-178 (2009).

Mahal, B.S., Clark, D.E.R. ve Simmons, J.E.L., "Geometric modeling of sheet deformation within a virtual environment", ***The International Journal Of Virtual Reality***, 5(1) (2001).

Michalewicz, Z., " Genetic Algorithms + Data Structures = Evolution Programs", ***Springer-Verlag***, Berlin, 1992.

Mitchell, M., "Introduction to Genetic Algorithms", ***Bradford Pres***, London, 10-48 (1998).

Provot, X., "Deformation Constraints in a Mass-Spring Model to Describe Rigid Cloth Behavior", ***Proc. Graphics Interface '95***, 147-154 (1995).

Rashedi, E., Nezamabadi, H. ve Saryazdi, S., "GSA: A Gravitational Search Algorithm", ***Information Sciences***, 179(13): 2232–2248 (2009).

Reeves, C.R., "Modern Heuristic Techniques for Combinatorial Problems", ***McGraw-Hill Book Company Inc***., Europe, 1995.

KAYNAKLAR (Devam Ediyor)

Serway, R.A. ve Beichner, R.J., "Fen ve Mühendislik İçin Fizik 1", Prof. Dr. Kemal Çolakoğlu, ***Palme Yayıncılık***, Ankara, 2007.

Shapiro, J., "Genetic Algorithms in Machine Learning", ***Machine Learning and Its Applications***, 2049: 146-168 (2001).

Storn, S. ve Price, K., "Differential Evolution - A simple and efficient adaptive scheme for global optimization over continuous spaces", ***ICSI Tech. Rep.***, TR-95-012, (1995).

Şekercioğlu, A.S., "Moleküler Modelleme İle Yumuşak Doku Deformasyonunun Modellenmesi Ve Simülasyonu", Yüksek Lisans Tezi, ***Dumlupınar Üniversitesi Fen Bilimleri Enstitüsü***, Kütahya, 2010.

Taghipour, M., Moradi, A. R. ve Yazdani-Asrami, M., "GSA Trained ANN for Educational Purposes", ***2010 IEEE Conference on Open Systems (ICOS 2010)***, Kuala Lumpur, Malaysia, (2010).

ÖZGEÇMİŞ

Kişisel Bilgiler

ADI SOYADI : Nihan KAZAK

DOĞUM YERİ VE TARİHİ : Denizli / 17.07.1987

Eğitim Durumu

Lisans Öğrenimi : Düzce Üniversitesi Bilgisayar Öğretmenliği

Bildiği Yabancı Diller : İngilizce

İş Deneyimi

Stajlar : Denizli Basma Sanayi A.Ş. Bilgi İşlem

Cafer Sadık Abalıoğlu Holding Bilgi İşlem

Çalıştığı Kurumlar : Bilecik Üniversitesi Mühendislik Fakültesi

İletişim

Adres: Bilecik Üniversitesi Gülümbe Kampüsü Mühendislik Fakültesi

Tel: 0228 216 01 01 - 1367

E-Posta Adresi: nihan.kazak@bilecik.edu.tr