T.C.
SAKARYA ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

PARÇACIK SÜRÜ OPTİMİZASYONU VE ÇOĞALAN SÜRÜ ALGORİTMASININ YÜZEY GERİ ÇATIMI PROBLEMİNDE UYGULANMASI VE KARŞILAŞTIRILMASI

YÜKSEK LİSANS TEZİ

Hüseyin DEMİRCİ

Enstitü Anabilim Dalı	**:**	**BİLGİSAYAR VE BİLİŞİM MÜHENDİSLİĞİ**
Tez Danışmanı	**:**	**Doç. Dr. Ahmet Turan ÖZCERİT**

Aralık 2014

T.C.
SAKARYA ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

PARÇACIK SÜRÜ OPTİMİZASYONU VE ÇOĞALAN SÜRÜ ALGORİTMASININ YÜZEY GERİ ÇATIMI PROBLEMİNDE UYGULANMASI VE KARŞILAŞTIRILMASI

YÜKSEK LİSANS TEZİ

Hüseyin DEMİRCİ

Enstitü Anabilim Dalı : **BİLGİSAYAR VE BİLİŞİM MÜHENDİSLİĞİ**

Bu tez 25/12/ tarihinde aşağıdaki jüri tarafından oybirliği / oyçokluğu ile kabul edilmiştir.

Doç. Dr. A. Turan ÖZCERİT	Doç. Dr. Cüneyt BAYILMIŞ	Yrd. Doç. Dr. Bilal Çobanoğlu
.....	.....	.....
Jüri Başkanı	**Üye**	**Üye**

TEŞEKKÜR

Yüksek lisans eğitimimin ders dönemi boyunca bana yardımcı olan Sayın Doç. Dr. Adem Alpaslan ALTUN'a, yüksek lisans eğitimimin tez dönemim boyunca bana yardımcı olan, bilgilerini esirgemeyen, yol gösteren ve sorunlarımı dinleyen ve sabırla çözüm üreten Sayın Doç. Dr. Ahmet Turan ÖZCERİT'e, tez ile ilgili çalışmam da bilgi ve birikimlerinden yararlandığım değerli hocalarım ve arkadaşlarıma teşekkürlerimi sunarım.

İÇİNDEKİLER

SİMGELER VE KISALTMALAR LİSTESİ

PSO	: Parçacık Sürü Optimizasyonu
ÇS	: Çoğalan Sürü
NURBS	: Tekdüze Olmayan Rasyonel B-Spline
EA	: Evrimsel Algoritma
BT	: Benzetilmiş Tavlama
ÇABT	: Çok Amaçlı Evrimsel Algoritma
GA	: Genetik algoritma
x_i	: Değişken
v_i	: Parçacık hızı
t, u, v	: Parametre
c_1, c_2	: Sosyal bileşen
w	: Parçacık hızı iç ağırlık bileşeni
ψ	: Üreme oranı
p_1	: Ebeveyn 1
p_2	: Ebeveyn 2
$N_{i,k}(t)$	: B-Spline temel fonksiyonu
$R_{i,k}(t)$	: NURBS temel fonksiyonu
B_i, $B_{i,j}$	: Kontrol noktası
P(t)	: Oluşturulan eğri
Q(u,v)	: Oluşturulan yüzey
$w_{i,j}$, w_i	: NURBS kontrol noktası ağırlık değeri
$u_{0...n}$, $v_{0...n}$	: Düğüm değerleri
E	: Toplam Hata

ŞEKİLLER LİSTESİ

TABLOLAR LİSTESİ

ÖZET

Anahtar kelimeler: Parçacık Sürü Optimizasyonu, Optimizasyon, Melez Algoritma, Yüzey Geri Çatımı

Optimizasyon işlemi bir şeyi en iyi hale getirmek anlamına gelmektedir. Günlük yaşantımızda birçok alanda farkında olmadan optimizasyon işlemini uygulamaktayız. Gerek yolculuk yaparken, gerekse bir şeyler satın alınırken bir maliyet ve kazanç kıyaslaması yapmaktayız. Günümüzde bilgisayarlarında hızlanması ile daha büyük optimizasyon problemleri bilgisayarlar yardımıyla çözülebilmektedir ve bunun için birçok algoritma geliştirilmiştir. Ancak geliştirilen bu algoritmalar mutlak çözümü bulmak için üzerinde çalıştırıldıkları bilgisayarlar ne kadar hızlı olurlarsa olsunlar çok süre harcamaktadırlar. Bu sorunun çözümü için doğadan ilham alan meta sezgisel algoritmalar geliştirilmiştir. Bu algoritmalar tam çözüm yerine, bu çözme en yakın sonucu elde etmektedirler ve daha hızlı çalışmaktadırlar. Buna rağmen halen birçok eksik yönleri vardır ve bu eksiklikleri gidermek için ilgili algoritmanın bazı adımları farklı algoritmalar ile birleştirilerek melez algoritmalar oluşturulmaktadır.

Bu tez çalışmasında, Parçacık Sürü Optimizasyonu (PSO) algoritması ile Çoğalan Sürü (ÇS) algoritmasının yüzey geri çatımı probleminin çözümü üzerindeki performansı karşılaştırılmıştır. Yüzey geri çatımı için tek düze olmayan rasyonel B-Spline (NURBS) parametrik yüzeyleri kullanılmıştır.

Yapılan çalışmada, PSO algoritması ÇS algoritmasına göre daha iyi bir sonuç sergilemiştir. Ancak döngü sınırının sonuna dek arama çeşitliliği açısından ÇS algoritmasının daha baskın olduğu gözlemlenmiştir. Algoritmaların parametre ayarlarına göre sonuçların önemli miktarda değiştiği gözlenmiştir.

APPLICATION AND COMPARISION OF THE PARTICLE SWARM OPTIMIZATION AND BREDING SWARMS ALGORITHM ON THE SURFACE RECONSTRUCTION PROBLEM

SUMMARY

Keywords: Particle Swarm Optimization, Optimization, Hybrid Algorithm, Surface Reconstruction

Optimization process is to find the desired solution for a predefined problem. In our daily life we usually use this procedure i.e. in trading, in travelling. With the benefit of the advances on computer technology, much bigger and complex optimization problems can be solved with computers and therefore, many algorithms have been developed. Although these algorithms focus on the finding the exact solutions, they spend too many computational time no matter how the computer fast. To solve this problem, nature inspired meta-heuristic algorithms were developed. These algorithms find the approximate solution instead of the exact solution and they run fast although they have many drawbacks. To eliminate these drawbacks, the steps of meta-heuristic algorithms are combined with other algorithms; hence, hybrid algorithms are created.

In this work, Breeding Swarms (BS) algorithm, which is a hybrid form of Particle Swarm Optimization (PSO) algorithm, has been compared with original algorithm on the efficiency of surface reconstruction problem's solution. NURBS surfaces are used on surface reconstruction problem.

The results obtained from this thesis suggest that PSO algorithm has been proved better results than BS algorithm. However, BS algorithm's diversity of the search process is more dominant than other is. The fact that the results vary in drastically in regard with the parameter settings of the algorithms employed.

BÖLÜM 1. GİRİŞ

1.1. Motivasyon

Optimizasyon süreci günümüzde hemen hemen her alanda uygulanmaktadır. Basit optimizasyon problemleri eskiden geleneksel yollarla çözülürken günümüzde zorlaşan optimizasyon problemleri bilgisayar yardımıyla çözülmektedir.

Optimizasyon problemlerinin çözümleri için birçok algoritma geliştirilmiştir. Her algoritmanın uygulanmak istenen problemin türüne göre pozitif ve negatif yönleri bulunmaktadır. Genel olarak literatürde optimizasyon algoritmalarının başarımlarını test etmek amacıyla birçok test problemi kullanılmıştır. Bu test problemleri birçok makalede kullanıldığından dolayı bir ölçüt haline gelmiştir. Ancak mesele gerçek dünya problemlerini çözmeye gelince, bu test problemleri üzerinde vaat edilen yenilikler ve başarımlar çoğunlukla aynı sonucu vermemektedir.

Bu tez çalışmasında Parçacık Sürü Optimizasyonu (PSO) algoritması ile Çoğalan Sürü (ÇS) algoritmasının yüzey geri çatımı probleminin çözümü üzerindeki performansı karşılaştırılmıştır. Yüzey geri çatımı için tek düze olmayan rasyonel B-Spline (NURBS) parametrik yüzeyleri kullanılmıştır.

Yüzey geri çatımı probleminin seçilmesinin nedeni günümüzde birçok alanda üç boyutlu modellerin kullanılmasıdır. Gelişen teknoloji ile her geçen gün bilgisayarların grafik işlemcileri (GPU) hızlanmaktadır. Bu sayede eskiden çok uzun süren modelleme işlemleri artık saniyeler içinde yapılmaktadır. Gelişen bu teknoloji ile piyasada daha iyi grafikler için talep artmıştır. Daha iyi grafikleri elde etmek için çözünürlüğün artması gerekmektedir. Buna bağlı olarak modellenen nesnenin veri boyutunun da artması söz konusudur. Modellenen nesnenin verilerinin bellek üzerinde doğrudan nokta bulutu şeklinde tutulması yerine parametrik biçimlerinin tutulması veri boyutunda azalmaya sebep olacaktır. Parametrik hale gelen modeller herhangi bir üç boyutlu model tasarım ve görüntüleme programı üzerinde kullanılabilir hale gelmiş olacaktır. Nokta bulutunun parametrik veriler haline getirilmesi

işlemi kısaca yüzey geri çatımı işlemi olarak tanımlanmaktadır. Piyasada hali hazırda bu işlemleri düşük çözünürlükteki yüzeyler için gerçekleştiren programlar bulunmaktadır.

1.2. Yapılan Çalışmalar

Daha önce yüzey geri çatımı problemi üzerinde evrimsel algoritmalar ile ilgili yapılan çalışmalarda, Weinert ve arkadaşları Evrimsel Algoritma (EA) ve Benzetilmiş Tavlama (BT) kullanarak 3 boyutlu nokta bulutlarından üçgenleme metoduyla 3 boyutlu yüzey geri çatımı işlemi yapılmıştır [1]. Weinert ve arkadaşları başka bir çalışmada ise 3 boyutlu veri noktası bulutu ilk önce katı yapısal geometri bileşenine dönüştürülüp ardından da bu dönüştürülen nokta kümesine NURBS yüzey uydurma işlemi evrimsel strateji ve genetik programlama algoritmaları kullanılarak oluşturulmuştur [2]. Wagner ve arkadaşları yaptıkları bir çalışmada Çok Amaçlı BT (ÇABT) kullanılarak NURBS yüzeyleri tekil değer ayrışımı yöntemi ile geri çatılmaya çalışılmıştır ve bu yöntemde veriler ön işleme ile azaltılmaya çalışılmıştır [3]. Sen ve Zheng yaptıkları bir çalışmada BT kullanılarak nokta bulutu ile optimum üçgenleme işlemi yapılmıştır ve yaklaşım işlemi amaç olarak kabul edilmiştir [4]. Goinski yaptığı bir çalışmada girilen 3 boyutlu nokta bulutu EA kullanılarak ilk önce bir çevreleyen kutu içerisine alınıp daha sonra her döngüde biraz daha detay eklenecek şekilde sarılmaya çalışılıp bir ızgara modeli oluşturularak yüzey geri çatılmak istenilmiştir [5]. Kodama ve arkadaşlarının yaptığı bir çalışmada ise elektron mikroskopundan taranan 3 boyutlu nokta kümesi örnekleri ile Genetik Algoritma (GA) kullanılarak üçgenleme ve elektron emisyonu tekniği kullanılarak yüzey geri çatımı işlemi yapmıştır ve sonuçlar BT ile karşılaştırılmıştır. Yine bu çalışmada veriler 2 boyutlu olarak ele alıp üçgenleme yaptıktan daha sonra z değerini hesaba katıp yaklaşım yapılmıştır [6]. Galvez ve Iglesias yaptıkları çalışmada PSO kullanarak 3 boyutlu nokta bulutları hiçbir önişleme yöntemi olmaksızın programa verilmiş ve yüzey geri çatımı işlemi yapılmıştır. Bu çalışmada PSO için yıldız topolojisi ve amaç fonksiyonunda tekil değer ayrışımı yöntemi kullanılmıştır [7]. Galvez ve Iglesias'ın yaptığı çalışma hariç diğer bütün çalışmalarda arzu edilmeyen derecede yüzeyler elde edilmiştir. Ancak Galvez ve Iglesias'ın yaptığı çalışmada büyük bir başarı elde edilerek girdi olarak alınan yüzeye çok yakın yüzeyler elde edilmiştir. Yapılan bu çalışmalarda yüzey geri çatımı işleminin zor bir optimizasyon problemi olarak kabul edilebileceği gözükmektedir.

1.3. Tezin Amacı

Tezin amacı, PSO ve ÇS algoritmalarının literatürde kullanılan test fonksiyonları üzerindeki çözüm bulma performansı ile yüzey geri çatımı probleminin çözümünü bulma performansları

arasında kıyaslama yaparak, test fonksiyonları ve algoritmaların başarımları üzerine değerlendirmelerde bulunmaktır.

1.4. Tez Planı

Tez planı olarak 2. bölümde optimizasyon ile ilgili genel kavramlar tanımlar ve problem sınıfları anlatılmıştır. 3. bölümde meta sezgisel algoritmalar anlatılmış olup PSO ve ÇS algoritmalarının akış şemaları, algoritmaları ve bu algoritmaların kullandığı yapısal denklemler verilmiştir. 4. bölümde parametrik eğriler ve yüzeylerin tanıtımı yapılmış ve bu konularla ilgili matematiksel bilgiler anlatılmıştır. 5. bölümde yüzey geri çatımı problemi, NURBS yüzeyleri üzerinde problemin uygulanması ve PSO algoritmasının bu probleme uygulanması anlatılmıştır. 6. bölümde ise yapılan uygulama sonuçları ve karşılaştırmalara yer verilmiştir. Son bölümde ise sonuç ve değerlendirme yapılmış, gelecekteki yapılabilecek çalışmalara yer verilmiştir.

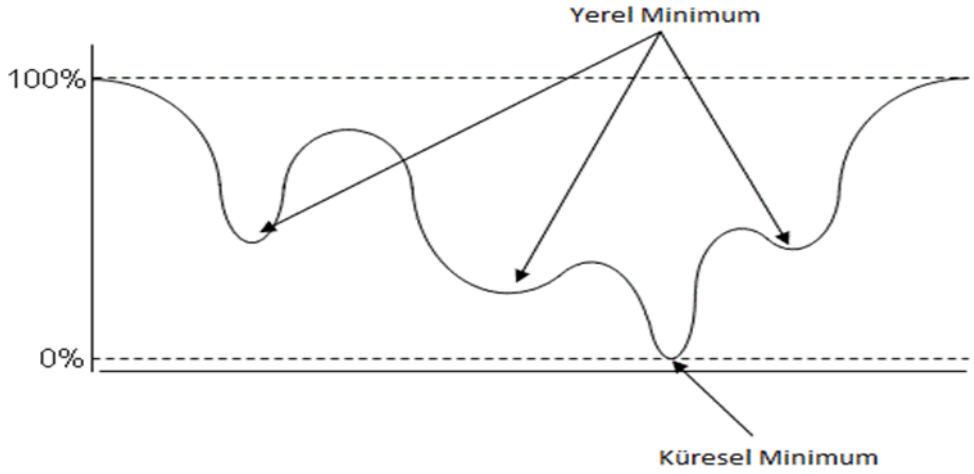
BÖLÜM 2. OPTİMİZASYON

2.1. Optimizasyonla İlgili Genel Kavramlar ve Tanımlar

Optimizasyon, tanımlanmış bir fonksiyonun belirli sınır aralıklarında minimum ya da maksimum noktasının ve bu noktayı sağlayan değişkenlerin bulunması demektir. Günlük hayatta optimizasyon işlemini farkında olmadan birçok işimizde kullanmaktayız. Bir otomobil ile minimum yakıt tüketecek şekilde istenilen yere varılması, en kısa yoldan istenilen yere ulaşılması, bir makinenin çalışma ve bakım zamanlarının ayarlanarak üretim veriminin artırılması gibi durumlar günlük hayatta kullanılan optimizasyon problemlerine örnek olarak verilebilir.

Bir optimizasyon probleminde minimize edilecek ise bir maliyet uygunluk fonksiyonu veya maksimize edilecek ise bir kâr uygunluk fonksiyonu ve o problemle ilgili sınırlamalar bulunmalıdır. Bir problemde birden fazla uygunluk fonksiyonu bulunabilir. Birden çok uygunluk fonksiyonuna sahip problemlere çok amaçlı optimizasyon problemi denmektedir. Sınırlamalar o fonksiyonun parametre değerlerinin istenilen değerler dışına çıkmasını engeller ve bu sınırlamalar karar parametrelerine bağlı olarak ifade edilir. Bu sınırlamalar eşitlikler veya eşitsizlikler şeklinde olabilir. Sınırlamalar dâhilinde tüm çözümleri kapsayan bölgeye uygun çözüm bölgesi denir.

Bir optimizasyon probleminde uygun çözüm bölgesi içinde şartları sağlayan tek bir nokta var ise bu nokta o problem için küresel minimum veya maksimumdur. Ancak bu bölge içinde olan ve şartları sağlayan birden çok nokta varsa bu noktalara da yerel minimum veya maksimum denir. Bu noktalardan problemim cinsine göre en büyük veya en küçük olanına da küresel minimum veya maksimum denir.



Şekil 2.1. Yerel ve küresel minimum noktalarını içeren grafik

Örnek olarak bir optimizasyon problemi, matematiksel biçimde;

$$f = \sum_{i=0}^{n} x_i^2 \qquad \textbf{(2.1)}$$

x değişkenler, n maksimum değişken sayısını ve f uygunluk fonksiyonu olmak üzere ifade edilebilir.

2.2. Optimizasyon Problemlerinin Sınıflandırılması

Optimizasyon problemlerinin sınıflandırılması problemle ilgili amaç fonksiyonunun ve problemin parametrelerinin türlerine göre yapılmaktadır.

Bir optimizasyon probleminde sınırlamalar yoksa bu tür problemlere sınırlamasız optimizasyon problemi, sınırlamalar varsa sınırlamalı optimizasyon problemi denir.

Amaç ve sınırlama fonksiyonları doğrusal ise bu problem doğrusal programlama problemi, bu amaç ve sınırlama fonksiyonlarından herhangi biri doğrusal değilse bu probleme doğrusal olmayan programlama problemi denir. Eğer fonksiyon değerleri negatif olmayan tamsayı değerleri alıyorsa bu türde problemlere de tamsayı programlama problemleri denir.

Ayrık objelerin en iyi şekilde sıraya konması, gruplanması veya seçilmesini içeren problemlere ayrık optimizasyon problemi denir. Bu tür probleme gezgin satıcı problemi örnek olarak verilebilir. Eğer amaç fonksiyonun değişkenleri bir değer aralığında sürekli değer alabilen değişkenler ise bu tür problemlere ise sürekli optimizasyon problemleri denir.

Problem sınıflamaları aşağıdaki Tablo 2.1'de özet halinde verilmiştir.

Tablo 2.1. Optimizasyon problemlerinin sınıflandırılması

Karakteristiği	Özelliği	Sınıflandırma
Tasarım değişkenlerinin sayısı	Bir	Tek değişkenli
	Birden fazla	Çok değişkenli
Tasarım değişkenlerinin türü	Sürekli	Sürekli
	Tamsayı	Tamsayı veya kesikli
	Sürekli ve Tamsayı	Karışık tamsayı
Hedef ve kısıtlayıcı fonksiyonlar	Doğrusal fonksiyon	Doğrusal
	Kuadratik fonksiyon	Kuadratik
	Doğrusal olmayan fonksiyon	Doğrusal olmayan
Problem tanımı	Kısıtlama mevcut	Kısıtlamalı
	Kısıtlama yok	Kısıtlamasız

BÖLÜM 3. META SEZGİSEL ALGORİTMALAR

Optimizasyon alanında gerçek dünya problemleri karmaşık ve çözümü zor olan problemlerdir. Bu tür karmaşık ve çözümü zor olan problemlerde genellikle tam sonuç elde etmek için kullanılan algoritmalar yürütme zamanı açısından yavaş ve sadece çözmek için tasarlandığı problemde işleyecek biçimde tasarlanırlar. Bu tür algoritmaları başka zor ve karmaşık problemlerde kullanmak imkânsızdır. Bu sebeple kesin çözümü vermeyen ve daha hızlı çalışan algoritmalar geliştirilmiştir. Bu algoritmalara sezgisel algoritmalar denir.

Sezgisel algoritmalar diğer algoritmalara göre daha hızlıdır ve çözüm uzayındaki bütün olasılıkları değerlendirerek en iyi çözüme yakın bir çözümü bulmayı amaçlarlar. Ancak en iyi çözümün bulunacağını hiçbir zaman garanti etmezler. Bu tür algoritmalar hızlı çalışmalarına rağmen geliştirilme aşamasında üzerinde kullanılacağı problemin bilgilerinden yararlanıldığı için uygulanıldığı probleme bağlı olan algoritmalardır ve bu algoritmalar klasik sezgisel algoritmalar olarak adlandırılır. Klasik sezgisel algoritmalara A* Araması, Demet Araması, Tırmanış Araması, En İyi Öncelikli Arama, Açgözlü En İyi Öncelikli Arama algoritmaları örnek olarak verilebilir.

Klasik sezgisel algoritmaların dışında probleme bağlı olmayan meta sezgisel algoritmalar vardır. Meta kelime anlamı olarak üst seviye anlamına gelir. Yani meta sezgisel algoritmaları, üst seviye sezgisel algoritmalar olarak değerlendirebiliriz. Bu algoritmalar genellikle doğadan ilham alınarak tasarlanmışlardır ve birçok probleme sadece amaç fonksiyonunu değiştirerek uygulanabilir. Meta sezgisel algoritmalar problem hakkında hiçbir bilgiye sahip olmadığı halde en uygun çözüm için en uygun değişken değerlerini verebildiği için bir nevi kara kutu olarak düşünülebilir. Meta sezgisel algoritmalara Genetik Algoritma, Karınca Kolonisi Optimizasyonu, Yapay Arı Kolonisi Optimizasyonu, Parçacık Sürü Optimizasyonu örnek olarak verilebilir.

3.1. Genetik Algoritmalar

Genetik algoritma, biyolojik süreç modellenerek fonksiyonları optimize eden evrimsel algoritmadır. Genetik algoritmada her bir birey gen olarak ifade edilir ve bu genlerin oluşturduğu topluluğa ise popülasyon denir. Popülasyonun uygunluk değeri, belirli kurallar dâhilinde maksimize veya minimize edilir. Her yeni nesil, rasgele bilgi değişimi ile oluşturulan diziler içinde hayatta kalanların birleştirilmesi ile elde edilmektedir [8].

Algoritma adımları aşağıdaki şekildedir.

1. Belirtilen sınır aralığında popülasyon boyutu adedince rastgele değerler üretilir.
2. Amaç fonksiyonuna göre bu bireylerin uygunluk değerleri hesaplanır.
3. Hesaplanan uygunluk değerlerine göre kötü olan bireyler yok edilir.
4. Yokedilen bireyler yerine yeni bireyler mevcut yokedilmemiş bireylerden çaprazlama yöntemiyle oluşturulur. Burada mevcut bireylerden seçim işlemi belirli yöntemlerle yapılmaktadır.
5. Ardından her birey için rastgele bir mutasyon değeri oluşturulur bu oluşturulan değer önceden belirlenmiş olan mutasyon oranı değerinden küçükse mutasyon işlemi yapılır.
6. Belirli bir durdurma kriteri sağlanıncaya kadar 2 ile 5. adım arası tekrar edilir.

3.2. Karınca Kolonisi Optimizasyonu

Karınca kolonisi optimizasyonu karıncaların yiyecek arama davranışından esinlenerek üretilmiş bir algoritmadır. Karıncalar yiyecek ararken hareket ettikleri yol üzerine feromon denilen bir salgı salgılarlar. Arkadan gelen diğer karıncalar feromon miktarının yoğunluğuna göre o yoldan gider ve yiyecek kaynağına ulaşır. Bu algoritma genelde birleşimsel optimizasyon problemleri için kullanılır [9].

Algoritma ise bu biyolojik temelden yola çıkılarak oluşturulmuştur. Birleşimsel bir problem olan gezgin satıcı problemi için algoritma şu şekilde çalışmaktadır. Algoritmada her bir bireyin belirli bir feromon salgılama miktarı vardır. Seçilecek mesafeye göre bu feromon miktarı bölünür ve o yola eklenir. O yolu seçen birey miktarı ne kadar çoksa o kadar fazla feromon o yolda birikir. Bir sonraki döngüde bireyler seçeceği yolları feromon miktarına göre belirler.

3.3. Yapay Arı Kolonisi Optimizasyonu

Arıların yiyecek arama davranışından esinlenerek üretilmiş bir optimizasyon algoritmasıdır. Bir arı kolonisinde üç tür arı bulunmaktadır. Bunlar işçi arılar, gözcü arılar ve kâşif arılardır. Kâşif arılar rastgele kaynak araştırması yapar. Gözcü arılar bulunan kaynaklardan hangisine gideceğini seçer ve o kaynağa gider kaynağa ulaştıktan sonra işçi arı olarak nitelendirilir [10].

Algoritmada ilk önce kâşif arılar rastgele olarak dağılırlar ve kaynak bulurlar bu kaynağın uygunluğu amaç fonksiyonunun değeridir. Bu kaynak uygunluğuna göre bir rulet tekerleği oluşturulur. Gözcü arılar rulet tekerleği seçimine göre kaynak seçerler ve o kaynak etrafında arama yaparlar. Kaynağını daha fazla geliştiremeyen arılar belirli bir limite göre tekrar kâşif arı olur ve rastgele yeni bir kaynak bulurlar. Kâşif arıların daha iyi kaynak bulma ihtimali işçi arılara göre daha yüksektir [11].

3.4. Parçacık Sürü Optimizasyonu

Parçacık sürü optimizasyonu 1995 yılında J. Kennedy ve R. Eberhart tarafından önerilmiş sosyal tabanlı, sürü zekâsına dayalı bir optimizasyon algoritmasıdır. Bu algoritma kuş sürülerinin yiyecek arama davranışlarından esinlenerek oluşturulmuştur [12].

Algoritmanın temel mantığı bir kuş sürüsünde kuşların bulduğu yiyecek kaynağını diğer kuşlara bildirmesi ve bu kuşların bu kaynaklardan en iyi olanına doğru hareket ederek arama yapmasıdır. Bu sayede neredeyse bütün arama uzayı araştırılarak en iyi çözüme yakın bir çözüm bulunabilir. Sürüdeki kuşların kendi hafızaları bulunmaktadır ve ayrıca en iyi yiyecek kaynağının yerini de bilmektedirler.
Parçacık sürü optimizasyonunda bireylerin yer değiştirmesi aşağıdaki formüle göre yapılmaktadır;

$$x_i(t+1) = x_i(t) + v_i(t) \qquad \textbf{(3.1)}$$

x_i, o anki bireyin değişkeninin değeri; v_i, hız vektörü; t ise o anki döngü sayısıdır.

Hız vektörü ise orijinal Parçacık sürü optimizasyonunda aşağıdaki şekilde hesaplanmaktadır.

$$v_{ij}(t+1) = v_{ij}(t) + c_1 r_{1j}(t) * \left(y_{ij}(t) - x_{ij}(t)\right) + c_2 r_{2j}(t) * \left(\hat{y}_j(t) - x_{ij}(t)\right) \qquad (3.2)$$

$v_{ij}(t)$ t sayılı döngüdeki j=1,…,n boyutunda i. parçacığın hızı, $x_{ij}(t)$ t sayılı döngüdeki j boyutunda i. parçacığın pozisyonu, $y_{ij}(t)$ j. boyutunda i. parçacığın o ana kadarki kendinin bulduğu en iyi kaynağın konumu, $\hat{y}_j(t)$ j. boyutunda o ana kadar ki bulunmuş en iyi çözümün konumu, c_1 ve c_2 sırasıyla kişisel ve sosyal öğrenim katsayısı, r_{1j} ve r_{2j} [0,1] aralığında üretilmiş rastgele bir sayıdır. Yukarıdaki formüle göre hareket eden kuş, hem kendi en iyi çözümü hem de küresel en iyi çözümün konumunu kullanarak arama uzayının ilgili boyutunda hareket eder. Hangi en iyi çözümü kullanacağı c_1 ve c_2 katsayılarına bağlıdır. Bu katsayılardan c_1 katsayısı yüksek seçilirse kendi en iyi bulduğu çözüm etrafında arama yapar buna bölgesel arama denir. Eğer c_2 katsayısı yüksek seçilirse küresel en iyi çözüm etrafında arama yapar buna ise küresel arama denir. c_1 ve c_2 katsayıları literatürde genellikle birbirine eşit sayılar seçilmiştir. Eberhart ve Shi yaptıkları bir çalışmada bu katsayılar için $c_1 = c_2 = 1.494$ olarak alınabileceğini göstermişlerdir [13].

Parçacık sürü optimizasyon algoritmasının genel adımları aşağıdaki şekildedir;

1. Popülasyon oluşturulur. Her bir parçacığın başlangıç değeri ve hızı rastgele olarak atanır.
2. Uygunluk değeri hesaplanır. Her bir parçacığın uygunluk değeri verilen amaç fonksiyonuna göre hesaplanır.
3. Parçacığın en iyi değeri bulunur. Bir önceki adımda hesaplanan uygunluk değeri parçacığın hafızasında bulunan en iyi kişisel değer (pbest) ile karşılaştırılır. Eğer bir önceki adımdaki bulunan sonuç mevcut "pbest" sonucundan daha iyi ise yeni sonuç "pbest" ile değiştirilir.
4. Küresel en iyi parçacık bulunur. İkinci adımda her bir parçacık için hesaplanan uygunluk değeri programın hafızasında tutulan küresel en iyi çözüm (gbest) ile karşılaştırılır. Eğer daha iyi bir sonuç varsa bu sonuç "gbest" ile değiştirilir. Karşılaştırma işlemi bütün parçacıklar için yapılır.
5. Her parçacığın hızı ve konumu ayarlanır. Denklem 3.2'deki formüle göre parçacığın hız değişkeni ayarlanır ve Denklem 3.1'deki formüle göre parçacığın konumu ayarlanır. Bu işlem her bir parçacık için ayrı ayrı yapılır.
6. Durdurma şartı veya şartları sağlanıncaya kadar 2 - 7 adımları arasındaki işlemler tekrar edilir.

Durdurma kriteri seçilirken iki önemli noktaya dikkat edilmelidir. Birinci olarak durdurma şartı algoritmanın erken yakınsamasına sebep vermemelidir çünkü bu durum sadece bölgesel en iyi noktanın bulunmasına sebep olur. İkinci olarak durdurma şartı uygunluk fonksiyonunun çok fazla hesaplanmasını sebep olursa araştırma hesaplama maliyeti artar ki bu durumunda engellenmesi gerekir. Aşağıda bazı durdurma kriterleri verilmiştir.

Algoritma,

1. Daha önceden belirlenmiş bir maksimum döngü sayısına ulaşılırsa,
2. İstenilen bir sonuç bulunmuşsa,
3. Belirli bir süre boyunca herhangi bir gelişim gösterilmiyorsa,

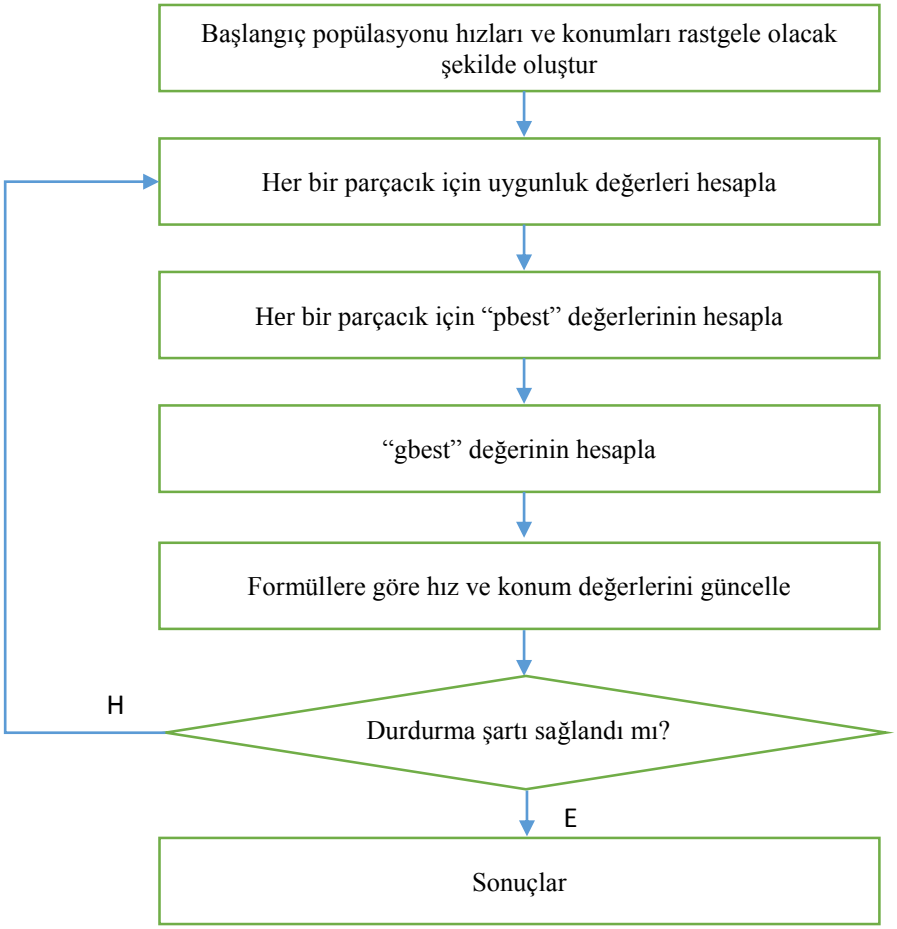
durdurulabilir. Algoritmanın akış diyagramı aşağıdaki Şekil 3.1.'de gösterilmiştir.

Algoritma zaman içinde birçok değişikliğe uğramıştır, birçok yeni bileşen eklenmiş ve bazı bileşenler değiştirilmiştir. Bu değişiklikler genellikle parçacıkların hızı ve konumunun değiştirilmesi aşamasına etki etmişlerdir. Daha sonraları Shi ve Eberhart hız değişkeninin bir iç ağırlık ile çarpılmasını öne sürmüşlerdir [14].

Bu durumda yeni hız değişimi formülü aşağıdaki şekilde tanımlanmıştır.

$$v_{ij}(t+1) = wv_{ij}(t) + c_1 r_{1j}(t) * \left(y_{ij}(t) - x_{ij}(t)\right) + c_2 r_{2j}(t) * \left(\hat{y}_j(t) - x_{ij}(t)\right) \qquad \textbf{(3.3)}$$

Yeni formüldeki w sabit bir değerdir ve literatürde ilk başta 0,9 değerini almasını önerilmiştir. Daha sonraları ise bu değer 0,9'dan başlayıp 0,4'e kadar doğrusal bir şekilde azalan bir değer olması önerilmiştir [13]. İç ağırlık ve kullanımı hakkında birçok strateji geliştirilmiştir. Bu stratejiler değerlendirildiğinde ortalama hatası en az olan strateji "kaotik iç ağırlık" stratejisi, en çok olan ise "kaotik rastgele iç ağırlık" stratejisidir. Ortalama döngü sayısında en iyi strateji "rastgele iç ağırlık" stratejisi en kötü olan strateji ise "sabit iç ağırlık" stratejisidir. En düşük hata bazında en iyi strateji "sabit iç ağırlık" ve "doğrusal azalan iç ağırlık" stratejisi, en kötü olan strateji ise "kaotik rastgele iç ağırlık" stratejisi ile "küresel-yerel en iyi iç ağırlık" stratejisi olarak belirlenmiştir [15]. Bu stratejilerin bazıları Tablo 3.1.'de verilmiştir.



Şekil 3.1. Parçacık sürü optimizasyonu algoritması akış şeması

PSO algoritmasının sözde kodu aşağıda verilmiştir.

```
n boyutlu S sürüsünü oluştur.
for i=0:n
        Si.x = random();
do
        for i=0:n
                f(Si.xi);
                if(f(Si.xi)<f(Si.yi)) Si.yi= Si.x;
                if(f(Si.xi)<f(S.ŷ)) S.ŷ= Si.x;
                Si.x= Si.x+ Si.v;
while(durdurma kriteri)
Sonuç= S.ŷ;
```

Şekil 3.2 PSO algoritmasının sözde kodu

Tablo 3.1. İç ağırlık stratejileri ve formülleri

Sıra No	Formül	Strateji Adı	Kaynak
1	$w = 0.7$	Sabit İç Ağırlık	[14]
2	$w_k = w_{max} - \frac{w_{max} - w_{min}}{iter_{max}} \times k$	Doğrusal Azalan İç Ağırlık	[16]
3	$w = 0.5 + \frac{Random()}{2}$	Rastgele İç Ağırlık	[17]
4	$w(t) = w_{min} + (w_{max} - w_{min}) \times \exp\left(-\left[\frac{t}{\left(\frac{MAXITER}{10}\right)}\right]\right)$	Doğal Üslü İç Ağırlık Stratejisi (e1-PSO)	[18]
5	$w(t) = w_{min} + (w_{max} - w_{min}) \times exp\left(-\left[\frac{t}{\left(\frac{MAXITER}{4}\right)}\right]\right)^2$	Doğal Üslü İç Ağırlık Stratejisi (e2-PSO)	[18]
6	$w = (w_{max} - w_{min} - d_1) \times \exp\left(\frac{1}{1 + \frac{d_2 t}{t_{max}}}\right)$	Üssel Azalan İç Ağırlık	[19]
7	$z = 4 \times z \times (1 - z)$ $w = (w_1 - w_2) \times \frac{MAXiter - iter}{MAXiter} + w_2 \times z$	Kaotik İç Ağırlık	[20]
8	$z = 4 \times z \times (1 - z)$ $w = 0.5 \times Random() + 0.5 \times z$	Kaotik Rastgele İç Ağırlık	[20]
9	$w_i = \left(1.1 - \frac{gbest_i}{pbest_i}\right)$	Küresel – Yerel En İyi İç Ağırlık	[21]

Parçacık hızı konusundaki bir diğer yenilik ise bir parçacığın hızının belirli bir aralıkta kısıtlanması konusunda yapılmıştır. Eberhart ve arkadaşlarının yaptığı bir çalışmada parçacığın hızının [-Vmax, Vmax] gibi bir aralıkta sınırlandırılması ve bu aralığının genellikle parçacığın maksimum konumunun 0.1 ile 1.0 katı arasında bir değer olarak seçilmesi gerektiği belirtilmiştir [22].

Temel parçacık sürü optimizasyonu üzerinde yapılmış olan diğer bir değişiklik ise parçacıkların birbiri ile olan iletişim yapısının şeklidir. Temel algoritmada bütün parçacıklar birbiri ile iletişimdedir. Bunun yanında tekerlek, halka ve rastgele kenar topolojisi de bulunmaktadır [23]

3.5. Optimizasyonun Algoritmalarının Melezleştirilmesi

Melezleştirme işlemi zeki sistemlerde gelişmekte olan bir alandır. Melezleştirme işleminin amacı farklı yaklaşımların iyi özelliklerini alıp bir yaklaşıma uygulayarak o yaklaşımın zayıf olduğu noktayı yok etmektir. Melezleştirme işlemi için bir veya birden fazla yaklaşım kullanılabilir. Melezleştirme işlem sonucunda ortaya eskisinden daha etkili yeni bir algoritma ortaya çıkar. Melezleştirme işlemi problemin yapısına göre veya genel olarak da yapılabilir.

3.5.1. Parçacık sürü optimizasyonu melezleştirme çeşitleri

Literatürde birçok algoritma parçacık sürü optimizasyonu ile melezleştirilmiştir. Bu bölümde bu algoritmalardan ve yapılan çalışmalardan kısaca bahsedilecektir.

Parçacık sürü optimizasyonu (PSO) algoritmasının verimliliğini arttırmak için genetik algoritmayı da (GA) dâhil edersek Evrimsel Algoritmaların (EA) kullanımı çok yaygındır. Mevcut popülasyonlar için formüller mevcut olduğundan bu iki yaklaşımın da melezleştirme işlemine uyarlanmalarına gerek kalmamaktadır.

İlk olarak PSO'ya uygulanan melezleştirme yöntemi kötü performans gösteren parçacıkların konumları ve hızları turnuva seçimi ilkesine göre düzenlenmiştir. Test sonuçlarında bu şekilde konum değiştirmek dört fonksiyondan üçünde başarı göstermiştir ancak PSO'nun sosyal metaforundan uzaklaşmıştır [24].

Diğer bir melez PSO ise NichePSO'dur [25]. Bu PSO melezi GCPSO'yu [26] kullanarak oluşturulmuştur. GA'nın tekniklerini kullanan bu algoritma alt sürü liderlerini ilk çalışma anında Kennedy'nin sade biliş modelini [27] kullanarak eğittikten sonra nişler tanımlanır ve alt sürü yarıçapı ayarlanır. Optimizasyon ilerledikçe parçacıklar ileride birleşecek olan alt sürülere teker teker katılırlar. Parçacık hızı minimize edildiğinde alt sürüler yakınsama işlemini yapmış olur. Bu metot her seferinde yakınsama yapsa da yazarlar sonuçların parçacıkların ilk dağılış yerlerine göre değiştiğini ve buna son derece bağımlı olduklarını belirtmişlerdir.

Bir diğer melez ise parçacıkların PSO, GA veya tepe tırmanma algoritmalarından hangisini kullanacaklarını kendileri seçtikleri "yaşam döngüsü" modelidir. Bu modelde her bir parçacığa hangi metodu kullanacağı yetkisi verilmiştir. Her bir parçacık kendileri için en

üretken metodu yine kendi deneyimlerine göre seçmektedirler. Bu deneyimler uygunluk fonksiyonunu hangi yöntemle daha iyiye getirildiği temeline dayanmaktadır. Seçim işlemi yapıldıktan sonra bireyler seçtikleri popülasyona katılır ve ona göre işlemlerini gerçekleştirirler. Testlerde bu yaklaşım olağanüstü başarı göstermiştir [28].

Gaussian mutasyonu hız ve konum güncelleme fonksiyonları ile birleştirilmiştir. Tek modlu ve çok modlu fonksiyonlarda test edilmiştir. GA ve PSO algoritmalarından daha başarılı sonuçlar elde etmiştir [29].

Yukarıda verilen örneklerden de anlaşılacağı gibi melezleme işlemi birçok şekilde, birçok algoritma ve birçok yöntem ile yapılabilmektedir. Günümüzde halen melezleme işlemi yapılmakta ve algoritmaların eksik tarafları başka algoritmaları kullanarak giderilmektedir.

3.6. Çoğalan Sürü Algoritması

Bu algoritma Matthew Settles ve Terence Soule tarafından geliştirilmiştir. Bir genetik algoritma ile parçacık sürü optimizasyonu algoritmasının melezlenmesi sonucu elde edilmiştir. Bu algoritmada melezlenen kısım algoritmanın işleyişi ve hız değişiminde kullanılan yeni çaprazlama operatörü olan Hız Temelli Ortalanmış Çaprazlama (VPAC) operatörünün kullanılmasıdır [30].

Algoritma çalışırken belirli bir adıma kadar temel PSO gibi çalışır. Uygunluk değerleri hesaplandıktan sonra parçacıklar sıralanır. 0 ve 1 aralığında tanımlanan bir sabit değer olan üreme oranı ψ değeri ve popülasyon boyutu N değerine göre elemanlar elenir. Geriye kalan elemanlardan bir seçim havuzu oluşturulur. Bu havuzdan turnuva seçimi ilkesine göre iki adet birey seçilir ve bu bireylerden çaprazlama ve mutasyon işlemleri ile yeni elemanlar oluşturulur. Oluşan yeni değerler elenen eleman değerlerine atanır. Her atamadan sonra seçim işlemi tekrarlanır. Elenen eleman kalmadıktan sonra elenmeyen elemanlar temel PSO algoritmasına göre hız ve konum değişimi yapılır ve bu işlemler durdurma kriteri sağlanıncaya kadar devam eder.

Bu algoritmada kullanılan çaprazlama operatörü aşağıdaki şekilde tanımlanmıştır.

$$c_1(x_i) = \frac{p_1(x_i) + p_2(x_i)}{2.0} - \varphi_1 p_1(v_i)$$
$$c_2(x_i) = \frac{p_1(x_i) + p_2(x_i)}{2.0} - \varphi_2 p_2(v_i) \qquad \textbf{(3.4)}$$

$c_1(x_i)$ ve $c_2(x_i)$ çocuk 1 ve 2'nin i. boyutundaki konumudur. $p_1(x_i)$ ve $p_2(x_i)$ ebeveyn 1 ve 2'nin i. boyuttaki konumlarıdır. $p_1(v_i)$ ve $p_2(v_i)$ ebeveyn 1 ve 2'nin i. boyuttaki hız vektörleridir. φ değeri [0.0, 1.0] aralığında rastgele üretilmiş bir sayıdır. Çocuklar ebeveynlerin hız vektörlerini ve pbest değerlerini 1.çocuk 1.ebeveynin, 2.çocuk 2.ebeveynin değerini alacak şekilde devralırlar. Algoritmanın akış diyagramı aşağıda Şekil 3,2'de verilmiştir.

Mutasyon işlemi olarak Gaussian mutasyonu kullanılmıştır. Bu mutasyon operatörü için döngü sayısına göre 1'den başlayıp 0,1'e kadar azalan bir varyans uygulanmıştır. Gauss dağılım formülü ve Gaussian mutasyon operatörü formülü aşağıda tanımlanmıştır.

$$g(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2} \quad \textbf{(3.5)}$$

Burada σ dağılımın varyansını, μ standart sapmasını ve x ise [0,1] aralığında üretilen rastgele bir sayıyı göstermektedir. Bilindiği üzere standart sapma değerinin karesi varyans değerine eşittir. Varyansın döngü sayısına göre azaltılması işlemi aşağıdaki formüle göre yapılmıştır.

$$\sigma = \frac{(1-0.1)*(\max iter - iter)}{\max iter} + 0.1 \quad \textbf{(3.6)}$$

Çoğalan Sürü algoritmasının genel adımları aşağıdaki şekildedir.

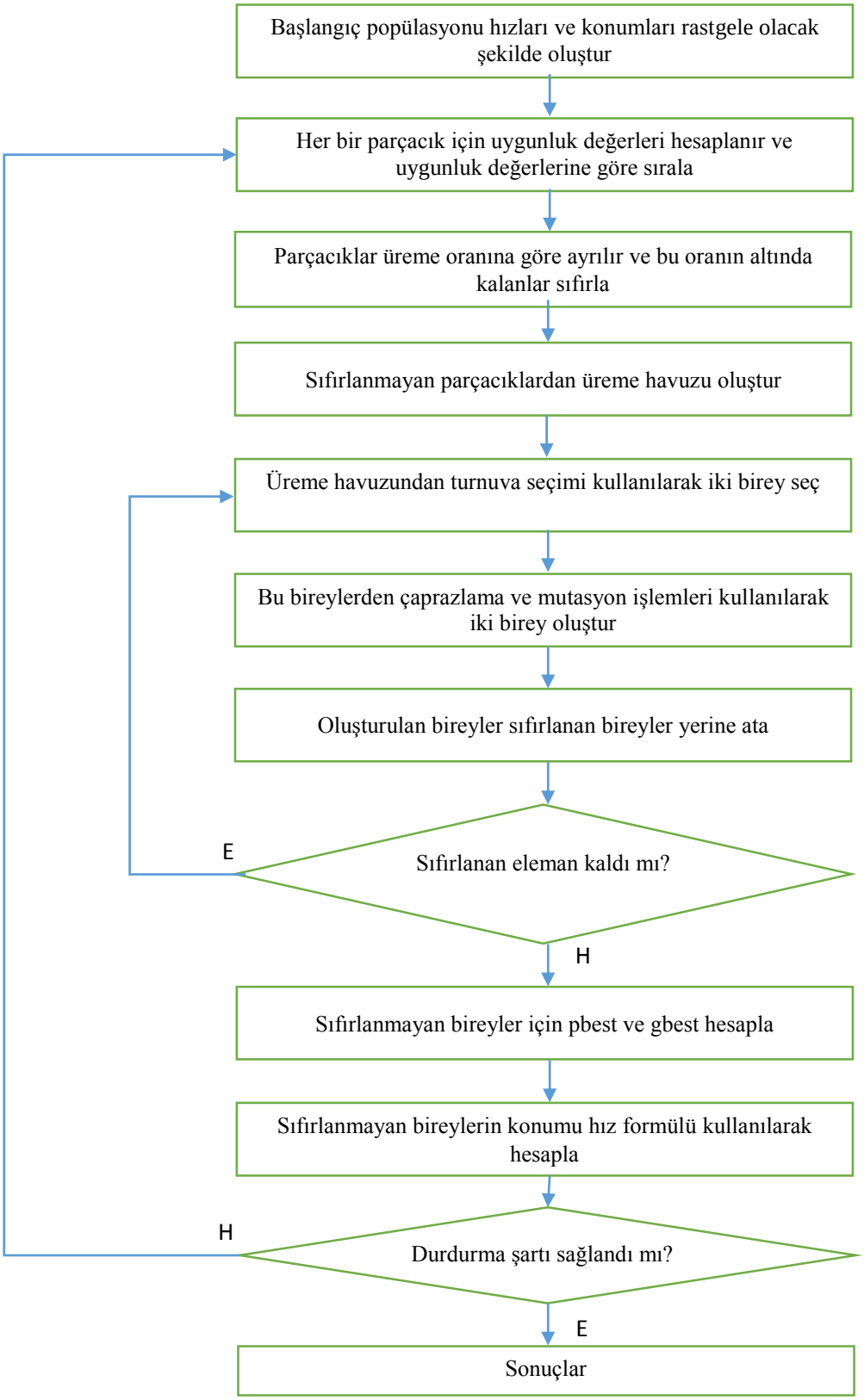
1. Popülasyon oluşturulur. Her bir parçacığın başlangıç değeri ve hızı rastgele olarak atanır.
2. Uygunluk değeri hesaplanır. Her bir parçacığın uygunluk değeri verilen amaç fonksiyonuna göre hesaplanır.
3. Sürü uygunluk fonksiyonu değerlerine göre sıralanır.
4. N×ψ ile hesaplanan eleman sayısından sonraki elemanların değerleri sıfırlanır. Bir nevi sürüden atılır.
5. N×ψ birey için geriye kalan N×(1-ψ) sayısınca bireye sahip olan havuzdan turnuva seçimi ile iki adet birey seçilir.
6. Seçilen elemanlardan iki adet yeni çocuk oluşturulur bu işlem Denklem 3.3'deki çaprazlama (VPAC) ve gaussian mutasyon işlemi ile yapılır.

7. Oluşturulan çocukların değerleri sürüden atılan elemanların değerleri yerine atanır ve atanan eleman sürüye tekrar kazandırılmış olur.
8. Her atama işleminden sonra 5. ve 6. adımlar tekrarlanır. Bu işlem sürüden atılan eleman kalmayıncaya kadar yapılır.
9. 10. , 11. ve 12. adımlar $N\times(1-\psi)$ parçacıklar için yani sürüden atılmayan parçacıklar için yapılır.
10. Parçacığın en iyi değeri bulunur. Bir önceki adımda hesaplanan uygunluk değeri parçacığın hafızasında bulunan en iyi kişisel değer (pbest) ile karşılaştırılır. Eğer bir önceki adımdaki bulunan sonuç mevcut "pbest" sonucundan daha iyi ise yeni sonuç "pbest" ile değiştirilir.
11. Küresel en iyi parçacık bulunur. İkinci adımda her bir parçacık için hesaplanan uygunluk değeri programın hafızasında tutulan küresel en iyi çözüm (gbest) ile karşılaştırılır. Eğer daha iyi bir sonuç varsa bu sonuç "gbest" ile değiştirilir. Karşılaştırma işlemi bütün parçacıklar için yapılır.
12. Her parçacığın hızı ve konumu ayarlanır. Denklem 3.2'deki formüle göre parçacığın hız değişkeni ayarlanır ve Denklem 3.1'deki formüle göre parçacığın konumu ayarlanır. Bu işlem her bir parçacık için ayrı ayrı yapılır.
13. Durdurma şartı veya şartları sağlanıncaya kadar 2 - 10 adımları arasındaki işlemler tekrar edilir.

ÇS algoritmasının sözde kodu aşağıda verilmiştir.

```
n boyutlu S sürüsünü oluştur. Ψ değerini belirle. n* Ψ boyutlu H havuz sürüsünü
oluştur.
for i=0:n
        Si.x = random();
do
        for i=0:n
                f(Si.xi);
                for j=n* ψ:n
                        Sj=null;
                for k=0: n* ψ
                        Hk=Sk;
                for l=n* ψ:n
                        Sl.x=(Hr1.x+Hr2.x)/2+Hr1.v;
                        Sl.x= Sl.x+GaussMutasyonu();
                if(f(Si.xi)<f(Si.yi)) Si.yi= Si.xi;
                if(f(Si.xi)<f(S.ŷ)) S.ŷ= Si.xi;
                Si.x= Si.x+ Si.v;
while (durdurma kriteri)
Sonuç= S.ŷ
```

Şekil 3.3 ÇS algoritmasının sözde kodu



Şekil 3.4. Çoğalan Sürü algoritmasının akış şeması

BÖLÜM 4. PARAMETRİK EĞRİLER VE YÜZEYLER

Yüzeyler eğrilerin çoğullaşmış halidir. Bu sebeple parametrik yüzeylerin anlaşılabilmesi için öncelikle parametrik eğrilerin anlaşılması gerekmektedir. Matematikte eğriler açık, kapalı ve parametrik biçimlerden biriyle ifade edilebilirler.

Bir eğri açık formda,

$$y = f(x) \tag{4.1}$$

şeklinde ifade edilirken kapalı formda,

$$\begin{aligned} f(x, y) &= 0 \\ f(x, y, z) &= 0 \end{aligned} \tag{4.2}$$

şeklinde ifade edilebilir. Bu iki gösterim şeklinin bilgisayar grafikleri ve bilgisayar destekli tasarım alanlarında çeşitli kullanımları olsa da eksene bağımlı gösterimler olduklarından dolayı çok değişkenli fonksiyonların temsilini düzgün bir şekilde sağlayamadıklarından pek tercih edilmezler.

Bir eğrinin örnek bir parametrik gösterimi aşağıda belirtilmiştir.

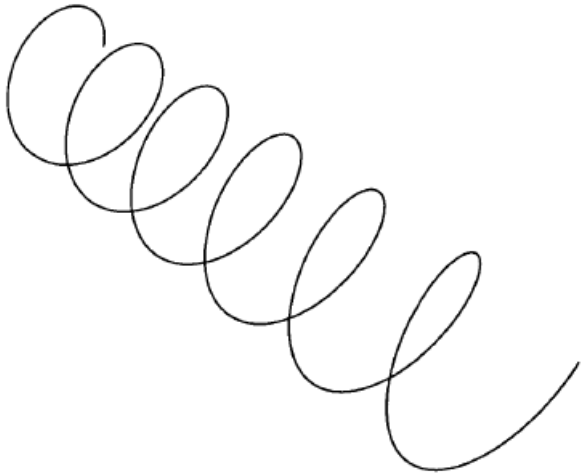
$$\begin{aligned} x &= f(t) \\ y &= g(t) \\ z &= h(t) \end{aligned} \tag{4.3}$$

Bu gösterimde t değişkeni parametredir ve oldukça esnektir. Parametrik gösterim eksenden bağımsız, çok değişkenli ve sonsuz türevi olan fonksiyonlarla kolayca uygulanabilirdir. Buradaki t parametresini sadece [0,1] aralığında tanımlanması gerekmemektedir.

Parametrik eğriler çok güçlü gözükmesine rağmen her zaman uygulanamazlar. İki parametrik eğrinin kesişime noktasının bulunması gibi bazı temel işlemleri yapmak diğer gösterimlere göre daha güçtür. Örnek bir parametrik sarmal ve fonksiyonu aşağıda verilmiştir.

$$\begin{aligned} x(t) &= r\cos(t) \\ y(t) &= r\sin(t) \\ z(t) &= at \end{aligned} \qquad \textbf{(4.4)}$$

Burada a ve r değişkenleri sıfıra eşit olmamakla birlikte parametre değeri olan t ise $-\infty \le t \le \infty$ aralığında tanımlıdır. x ve y eksenlerinde her 2π değerinde bir tam bir daire elde edilirken z yönünde ise $2\pi|a|$ kadar artış yada azalış olmaktadır. Buna bağlı olarak oluşan sarmal z yönünde uzanmış olmaktadır.



Şekil 4.1. r=1 a=0,7 ve $0 \le t \le 12\pi$ değerleri için oluşturulmuş dairesel sarmal [31]

4.1. Parametrik Eğriler

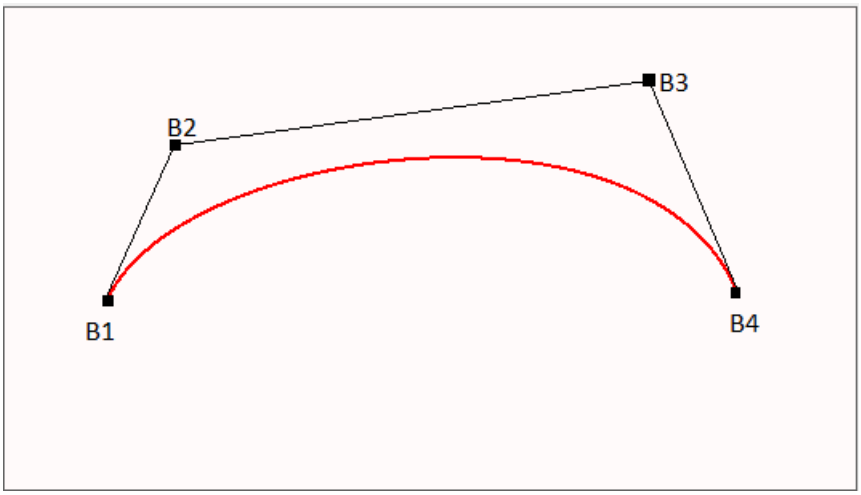
4.1.1. Bézier eğrileri

Bu eğri tipi ilk olarak Renault'da çalışan Fransız mühendis Pierre Bézier tarafından geliştirilmiştir. Bu eğrileri kullanarak sadece araba gövde parçalarını değil uçak kanatları, yat gövdelerini ve Fransız raylı sistemindeki koltukları tanımlamıştır [31].

Bézier eğrilerinin derecesinin temeli eklenen kontrol noktaları ile doğrudan bağlantılıdır. Bunun sebebi hesaplamada kullanılan Bézier temel fonksiyonu eğrinin birçok özelliğini belirlemesinden kaynaklanmaktadır. Bu fonksiyon Bernstein temel fonksiyonu olarak da bilinir.

Bu fonksiyonun bazı temel özellikleri aşağıda verilmiştir.

- Temel fonksiyondaki değerler gerçek sayı değerleridir.
- Eğrinin derecesi kontrol noktası sayısının bir eksiğine eşittir.
- Eğri genellikle kontrol noktalarının oluşturduğu şekli izler.
- Sırasıyla eğrinin ilk ve son noktaları ile ilk ve son kontrol noktaları birbiriyle kesişir.
- Eğri kontrol noktaları tarafından tanımlanan dış bükey omurga içerisinde kalır. Şekil 3.2'de bu özelliği görebilirsiniz.
- Eğri, kontrol noktalarına göre daha düz bir karakteristik izlemez.



Şekil 4.2. Bézier eğrisinin dış bükey omurga özelliği

Şekil 4.2'de B1, B2, B3 ve B4 noktaları kontrol noktalarıdır ve kırmızı renk ile çizilen eğri bu noktalara göre üretilmiştir.

Matematiksel olarak parametrik Bézier eğrisi aşağıda tanımlanmıştır.

$$P(t) = \sum_{i=0}^{n} B_i J_{n,i}(t) \qquad 0 \le t \le 1 \qquad \textbf{(4.5)}$$

ve Bernstein yada Bézier temel fonksiyonu ise

$$J_{n,i}(t) = \binom{n}{i} t^i (1-t)^{n-i} \qquad (0)^0 \equiv 1 \qquad \textbf{(4.6)}$$

$$\binom{n}{i} = \frac{n!}{i!(n-i)!} \qquad 0! \equiv 1 \qquad \textbf{(4.7)}$$

şeklinde tanımlanmıştır. Burada n eğrinin derecesini, B_i kontrol noktasını, $J_{n,i}(t)$ Bernstein ya da Bézier temel fonksiyonunu ve t ise parametreyi ifade etmektedir. Eğri derecesi kontrol noktası sayısının bir eksiği kadardır.

Bir eğrinin esnekliği derecesi ile bağlantılıdır. Bézier eğrilerinde eğri derecesi kontrol noktası ile doğrudan ilişkili olduğundan diyebiliriz ki Bézier eğrilerinin esnekliği kontrol noktası sayısına bağlıdır. Bu tip eğrilerin esnekliklerini arttırmak için daha fazla kontrol noktası eklemek gerekir.

4.1.2. B-Spline eğrileri

Bézier eğrileri her ne kadar iyi bir gösterim şekli gibi gözükse de iki nokta açısından kısıtlama oluşturmaktadır. Bunlardan birincisi eğrinin derecesini doğrudan doğruya kontrol noktaları ile bağlantılı olması ikincisi ise Bézier temel fonksiyonun küresel tabanlı olmasıdır.

Eğrinin derecesinin sürekli kontrol nokta sayısına bağlı olması kübik bir eğri elde edilmek istenildiğinde en fazla dört adet kontrol noktası girilmesi ve daha fazla noktaya izin vermemesi sonucu doğurmaktadır. Altı adet kontrol noktasına sahip bir Bézier eğrisi daima beşinci dereceden bir eğri olmak zorundadır bu durum ise esnekliği kısıtlamaktadır.

Bézier temel fonksiyonunun küresel tabanlı olması oluşturulan eğrinin kesintisiz olmasına sebep olmaktadır. Kontrol noktalarında yapılan ufak bir değişiklik temel fonksiyon yüzünden

bütün eğriye etki etmektedir ve bu sebeple eğri üzerinde yerel bir değişiklik yapmak imkânsız olmaktadır.

Bu sorunları aşmak için B-Spline (Basis Spline) temel fonksiyonunu kullanan B-Spline eğrileri üretilmiştir. Bu temel fonksiyon Bézier temel fonksiyonunu içerisinde özel bir durum olarak barındırmaktadır. B-Spline teorisi ilk olarak Schoenberg [33] tarafından ortaya atılmıştır. Bu temel fonksiyonun hesaplanmasındaki rekürsif tanımı bağımsız olarak Cox [34] ve de Boor [35] tarafından bulunmuştur. Riesenfield [36] ve Gordon [37] ise bu fonksiyonu B-Spline eğri tanımı üzerinde kullanmıştır.

B-Spline eğrilerinin genel matematiksel tanımı ve bağımlılıkları aşağıda verilmiştir.

$$P(t)=\sum_{i=0}^{n} B_i N_{i,k}(t) \quad t_{\min} \le t \le t_{\max} \;\; 2 \le k \le n+1 \tag{4.8}$$

Bu denklemde B_i n+1 adet kontrol noktasını, $N_{i,k}(t)$ B-Spline temel fonksiyonunu ve t ise parametre değerini ve k ise eğri derecesini bir fazlasını temsil etmektedir. B-Spline temel fonksiyonu olan $N_{i,k}(t)$'nin tanımı aşağıda verilmiştir.

$$N_{i,1}(t)=\begin{cases} x_i \le t < x_{i+1} & ise \quad 1 \\ \text{farklı durumlarda} & 0 \end{cases}$$
$$N_{i,k}(t)=\frac{(t-x_i)N_{i,k-1}(t)}{x_{i+k-1}-x_i}+\frac{(x_{i+k}-t)N_{i+1,k-1}(t)}{x_{i+k}-x_{i+1}} \tag{4.9}$$

Yukarıdaki Denklem 4.9'da x_i mevcut düğüm değeri ve t ise parametre değeridir. Düğüm sayısı(m) ise kontrol noktası sayısı(n) ve eğri derecesinin bir fazlasının(k) toplamına eşittir. Düğüm vektörleri genellikle [0,1] tanım aralığındadır. Düğüm vektörleri azalmayan sırada girilmelidir. Yani eklenen düğüm vektörü bir önceki düğüm vektöründen küçük olmamalıdır.

Düğüm yapısı ve temel fonksiyonun yapısı sayesinde kontrol noktaları bölgesel olarak etki etmektedir. Düğümlerin ekleniş yapısına göre B-Spline eğrileri kategorilere ayrılmaktadır.

Bir düğüm dizisinde eğer düğümler eşit aralıklarla girilmişse bu tip düğümlere tekdüze (uniform) düğüm denir. Eğer sonda ve başta k adet düğüm kendini tekrar ediyorsa bu tip

düğüm dizisine ise açık tekdüze düğüm denir. Tekdüze olmayan (non-uniform) düğümlerde ise başta ve sonda k adet düğüm tekrar edebilir ayrıca ortada olan düğümler ise eşit olarak artmayabilir veya kendi aralarında tekrar edebilirler. Buna örnek olan düğümler aşağıda verilmiştir [31].

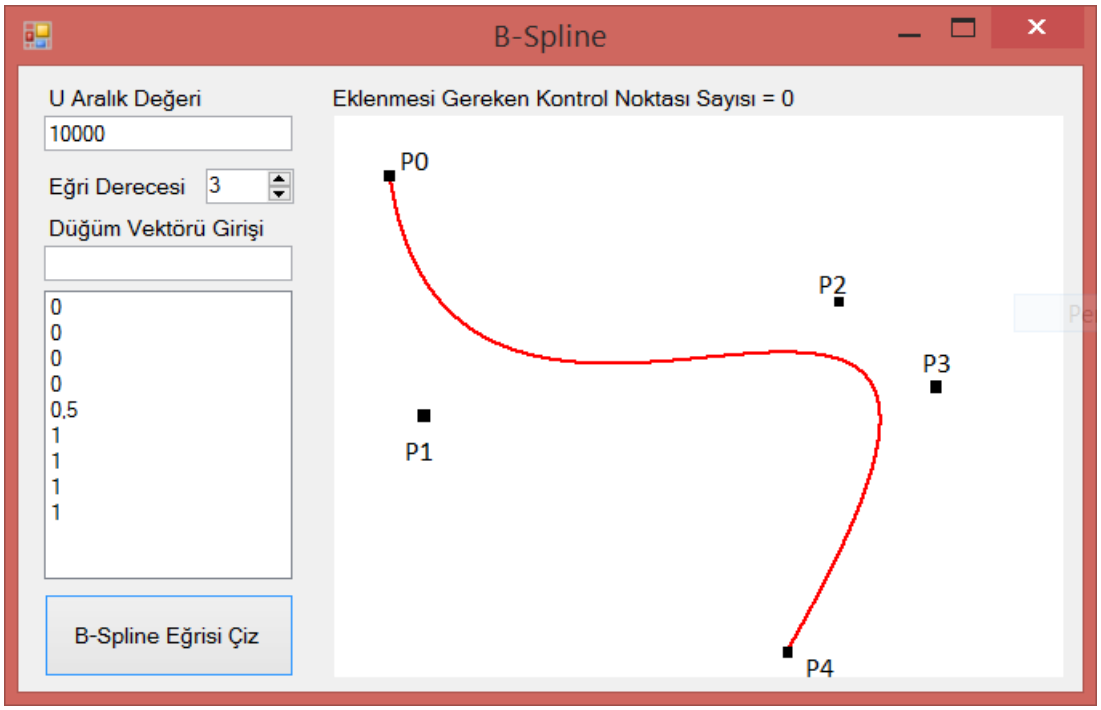
$$k=3\left[\begin{array}{cccccccc}0 & 0 & 0 & \frac{1}{3} & \frac{2}{3} & 1 & 1 & 1\end{array}\right]$$ Tekdüze Açık Düğüm Serisi

$$k=3\left[\begin{array}{cccccccc}0 & 0 & 0 & \frac{1}{2} & \frac{1}{2} & 1 & 1 & 1\end{array}\right]$$ Tekdüze olmayan Açık Düğüm Serisi

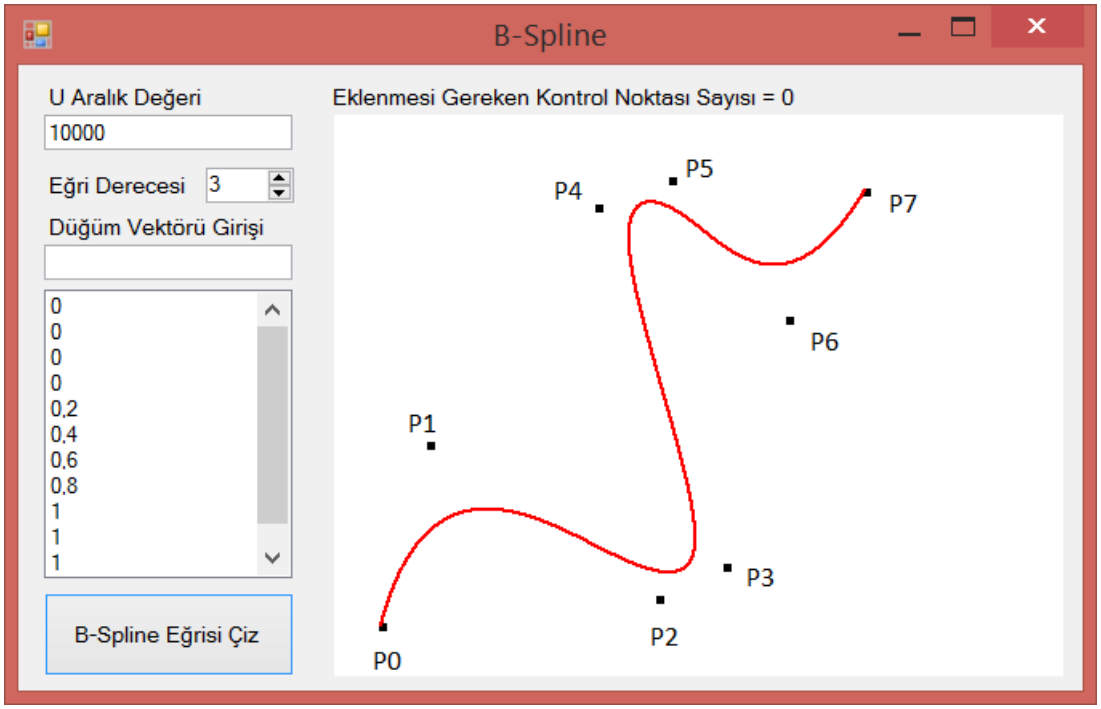
$$k=3\left[\begin{array}{cccccccc}0 & \frac{1}{7} & \frac{2}{7} & \frac{3}{7} & \frac{4}{7} & \frac{5}{7} & \frac{6}{7} & 1\end{array}\right]$$ Tekdüze Kapalı Düğüm Serisi

Düğümler oluşacak olan eğrinin karakteristiğini belirler. Eğer düğüm k adet başta tekrar ediyorsa eğri eklenen ilk kontrol noktasından başlar. Aynı şekilde eğer sonda k adet düğüm tekrar ediyorsa eğri eklenen son kontrol noktasında biter. Tekrar eden düğümlerde eğri ilgili kontrol noktasına doğru daha fazla yaklaşır.

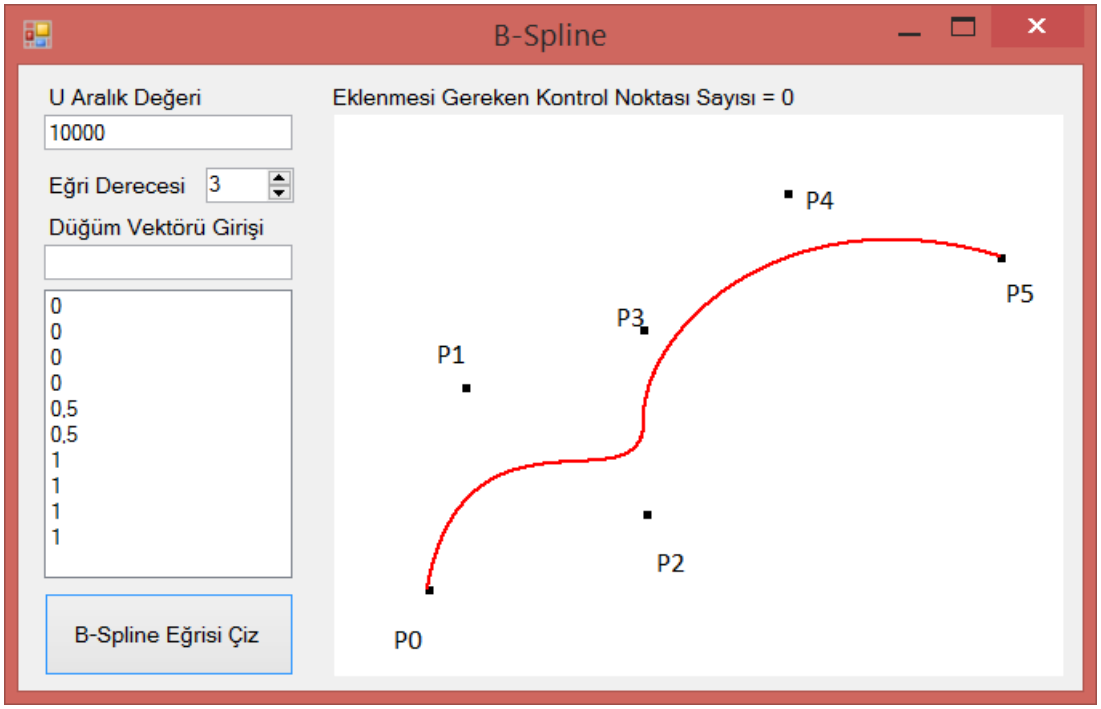
Aşağıda Şekil 4.3, 4.4 ve 4.5'te örnek B-Spline eğrileri ve düğüm değerleri verilmiştir.



Şekil 4.3. B-Spline eğrisi (Derece 3, Tekdüze, Açık)



Şekil 4.4. B-Spline eğrisi (Derece 3, Tekdüze, Açık)



Şekil 4.5. B-Spline eğrisi (Derece 3, Tekdüze Olmayan, Açık)

B-Spline eğrileri Bézier eğrilerinin tüm özelliklerine sahiptir. Ayrıca ek olarak eğrinin esnekliği kontrol noktası sayısına bağlı değildir. Bu sayede istenilen derecen eğri istenilen şekilde olabilmektedir. Eğer B-Spline eğrisindeki kontrol noktası sayısı, eğri derecesinin bir

fazlası kadar ise ve düğümler periyodik olmayan şekilde ise bu eğri kısaca Bézier eğrisi olarak tanımlanır.

4.1.3. Tekdüze olmayan rasyonel B-Spline eğrileri (NURBS)

NURBS eğrileri bilgisayar destekli tasarım ve bilgisayar destekli üretim alanlarında bir standart haline gelmiştir. Birçok değişkene sahip olması NURBS eğrilerini daha esnek ve bağımsız yapmıştır. Bu eğri çeşidi içerisinde B-Spline eğri çeşidini barındırmaktadır. Bu sebeple B-Spline eğrilerinin tüm özelliklerini miras alır. B-Spline eğrilerindeki yapıya ek olarak NURBS eğrilerinde her kontrol noktasına karşılık gelen [0,1] aralığında tanımlanmış bir ağırlık faktörü eklenmiştir. Bu sayede her bir kontrol noktasının eğri üzerindeki etkisi arttırılıp azaltılabilmektedir [31].

NURBS eğrileri rasyonel B-Spline eğrilerinin tek düze olmayan formudur. Rasyonel B-Spline'lar ise mevcut koordinat sistemin bağımsız olarak dördüncü boyutta yazılan parametrik eğri denklemleridir. Daha sonra bu denklemler 3 boyuta indirgenerek çizilebilir eğriler haline gelmektedir. Aşağıda genel dördüncü boyuttan olan NURBS eğrilerinin denklemi denklem 3.10'da ve bu denklemin nasıl tekrar 3.boyuta döndürüldüğü Denklem 4.11'de gösterilmiştir.

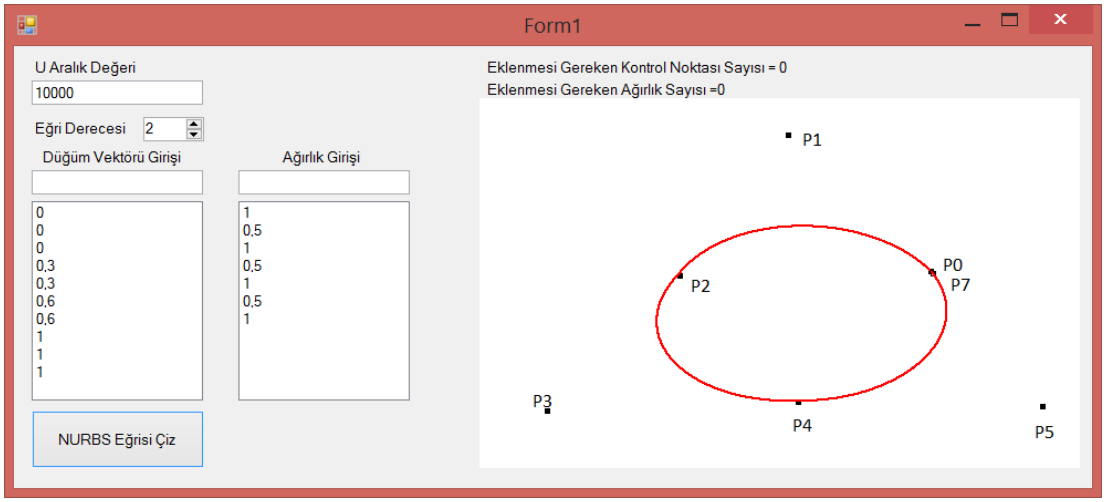
$$P(t) = \sum_{i=0}^{n} B_i^h N_{i,k}(t) \tag{4.10}$$

$$P(t) = \frac{\sum_{i=0}^{n} B_i h_i N_{i,k}(t)}{\sum_{i=0}^{n} h_i N_{i,k}(t)} = \sum_{i=0}^{n} B_i R_{i,k}(t) \tag{4.11}$$

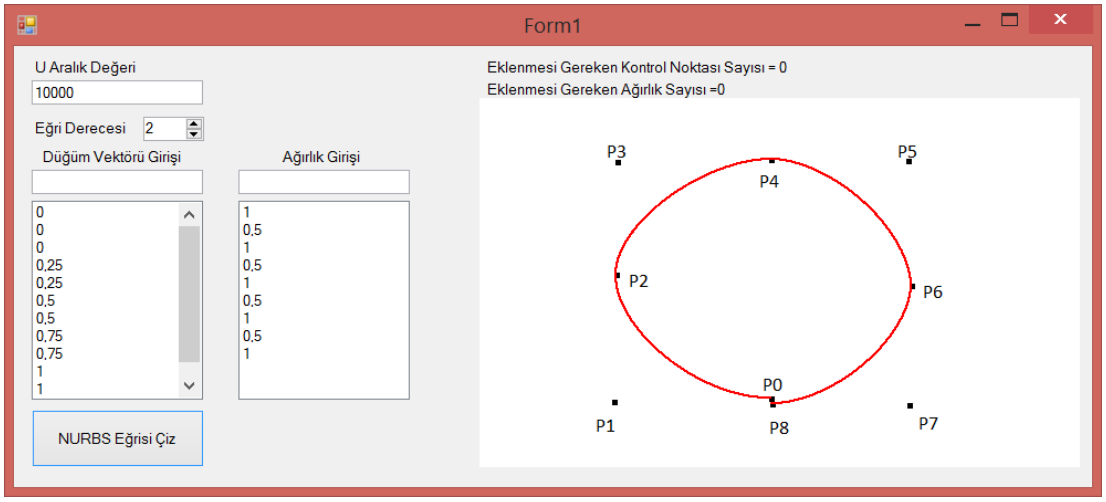
$$R_{i,k}(t) = \frac{h_i N_{i,k}(t)}{\sum_{i=0}^{n} h_i N_{i,k}(t)} \tag{4.12}$$

B_i^h dört boyutlu kontrol noktası, $N_{i,k}(t)$ daha önce açıklanan B-Spline temel fonksiyonu, B_i üç boyutlu kontrol noktasıdır ve h_i karşılık gelen noktanın ağırlık değeridir. Denklemde i'nin her değeri için $h_i \geq 0$ olarak alınmaktadır. $R_{i,k}(t)$ ise rasyonel B-Spline temel fonksiyonudur. $h_i = 1$ olduğu durumda bu denklem rasyonel B-Spline eğrilerini temsil etmektedir. Yine bu formdayken kontrol noktası sayısı eğri derecesinin bir eksiği olarak

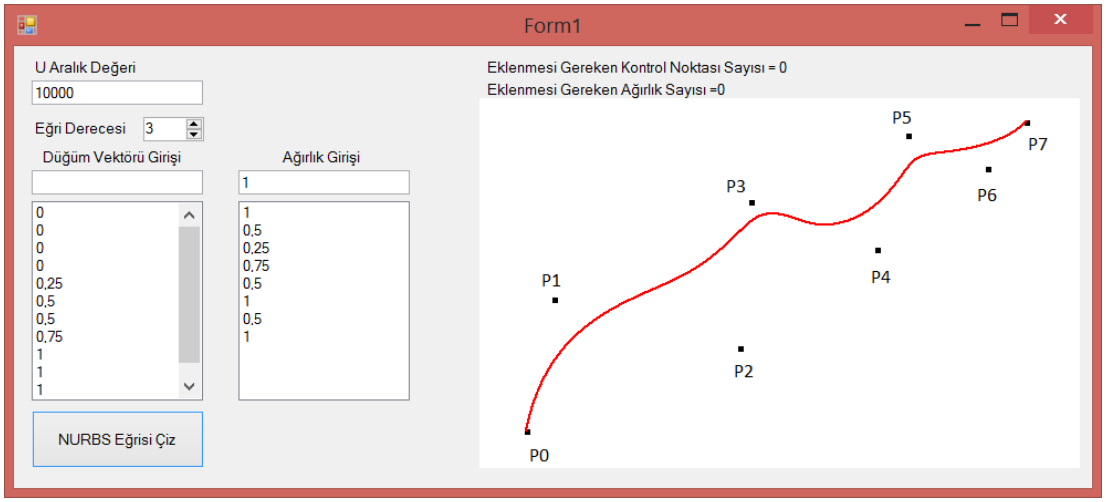
alınırsa Bézier eğrilerine ulaşılmış olur. Aşağıdaki şekil 4.6, 4.7 ve 4.8'de NURBS eğrilerine örnekler verilmiştir.



Şekil 4.6. NURBS eğrisi (derece 2)



Şekil 4.7. NURBS eğrisi (derece 2)



Şekil 4.8. NURBS eğrisi (derece 3)

Yukarıdaki şekil 4.8'de görüldüğü üzere eğri ağırlığı yüksek olan kontrol noktasına daha çok yaklaşmıştır ve daha düşük olan kontrol noktasına ise daha az yaklaşmıştır. Şekil 4.6 ve 4.7'de görüldüğü üzere eğri ağırlığı 1 olan kontrol noktasından geçmiştir.

NURBS eğrilerini birçok köklü bilgisayar destekli tasarım ve dijital eğlence firması esnekliği ve kullanımı kolay olduğu için standart olarak kabul etmiştir.

4.2. Parametrik Yüzeyler

4.2.1. Bézier yüzeyleri

Bézier yüzeyleri, Bézier eğrilerinin iki parametre uzayında genişletilmesiyle oluşturulur. Herhangi bir yüzey oluşturmak için sınırları belirleyecek olan en az dört adet kontrol noktasına ihtiyaç vardır. Bézier eğrilerinin bütün özelliklerini ve kısıtlarını Bézier yüzeyleri de Bernstein veya Bézier temel fonksiyonunu kullandığından dolayı miras olarak almaktadır [31].

Genel olarak bir Bézier eğrisinin matematiksel denklemi aşağıda verilmiştir.

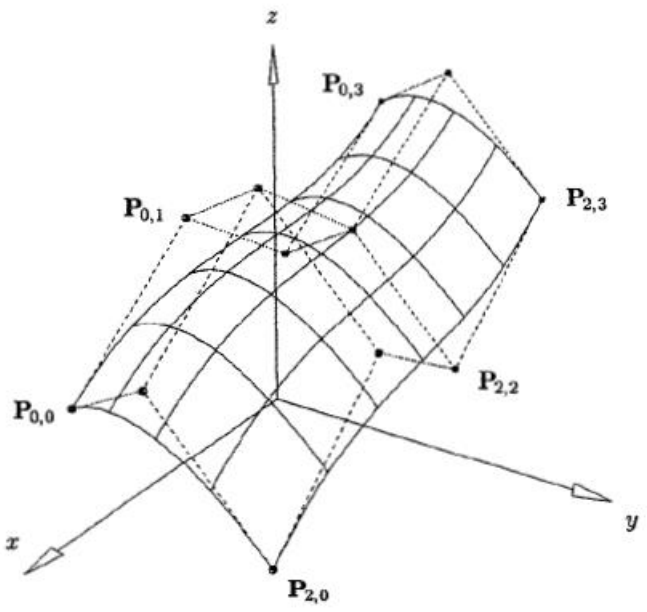
$$Q(u,v) = \sum_{i=0}^{n}\sum_{j=0}^{m} B_{i,j} J_{n,i}(u) K_{m,j}(v) \qquad \textbf{(4.13)}$$

Bu denklemde $B_{i,j}$ kontrol noktalarını, $J_{n,i}(u)$ ve $K_{m,j}(v)$ Bézier temel fonksiyonunu u ve v parametreyi, u yönündeki kontrol noktası sayısını n ve v yönündeki kontrol noktası sayısını m ifade etmektedir. Denklem 4.14 ve 4.15'te temel fonksiyonların tanımları verilmiştir.

$$J_{n,i}(u) = \binom{n}{i} u^i (1-u)^{n-i}$$
$$\binom{n}{i} = \frac{n!}{i!(n-i)!} \qquad \textbf{(4.14)}$$

$$K_{m,j}(u) = \binom{m}{j} w^j (1-w)^{m-j}$$
$$\binom{m}{j} = \frac{m!}{j!(m-j)!} \qquad (0)^0 \equiv 1 \qquad 0! \equiv 1 \qquad \textbf{(4.15)}$$

Kontrol noktası ağında her sütunda aynı sayıda kontrol noktası bulunmalıdır. Örnek bir Bézier yüzey örneği ve kontrol ağı aşağıda Şekil 4.9'da verilmiştir. Şekilde kontrol noktası ağının dışbükey omurgası içindeki yüzey görülmektedir.



Şekil 4.9. Bézier yüzeyi (12 adet kontrol noktası) [32]

4.2.2. B-Spline yüzeyleri

B-Spline yüzeyleri yine aynı şekilde B-Spline eğrilerinin iki parametre uzayında genişletilmesiyle oluşturulmuştur. B-Spline yüzeyleri B-Spline eğrilerinin bütün özelliklerini taşımaktadır. B-Spline yüzeylerinde iki parametre doğrultusunda da aynı tipte düğüm yapısı kullanmak zorunlu değildir [31].

B-Spline yüzeylerinde, yüzeyin şeklini ve karakteristiğini parametre doğrultusunda girilen düğümler belirler. Yüzeyin genel şekli ise kontrol noktaları tarafından belirlenir. Aşağıda denklem 4.16'da B-Spline yüzeylerinin genel tanımı verilmiştir.

$$Q(u,w)=\sum_{i=0}^{n}\sum_{j=0}^{m}B_{i,j}N_{i,k}(u)M_{j,l}(v) \qquad \textbf{(4.16)}$$

Bu denklemde $N_{i,k}(u)$ ve $M_{j,l}(v)$ B-Spline temel fonksiyonlarını, $B_{i,j}$ kontrol noktalarını, u ve v parametreyi ve m ile n ise sırasıyla u ve v yönündeki kontrol noktası sayılarını temsil etmektedir.

$$N_{i,1}(u)=\begin{cases} x_i \le u < x_{i+1} & ise \quad 1 \\ \text{farklı durumlarda} & 0 \end{cases} \qquad \textbf{(4.17)}$$

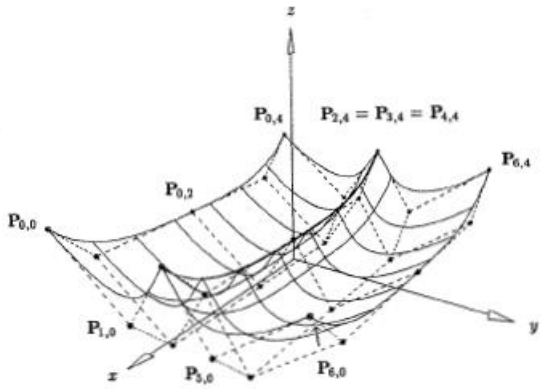
$$N_{i,k}(u)=\frac{(u-x_i)N_{i,k-1}(u)}{x_{i+k-1}-x_i}+\frac{(x_{i+k}-u)N_{i+1,k-1}(u)}{x_{i+k}-x_{i+1}}$$

$$M_{j,1}(v)=\begin{cases} y_j \le v < y_{j+1} & ise \quad 1 \\ \text{farklı durumlarda} & 0 \end{cases} \qquad \textbf{(4.18)}$$

$$M_{j,l}(v)=\frac{(v-y_j)M_{j,l-1}(v)}{y_{j+l-1}-y_j}+\frac{(y_{j+l}-u)M_{j+1,l-1}(u)}{y_{j+l}-y_{j+1}}$$

Denklemden de anlaşılacağı üzere her iki parametre yönünde de farklı adette kontrol noktası kullanılabilmekte ve buna bağlı olarak farklı adette düğüm ve farklı eğri dereceleri kullanılabilmektedir. Bu sayede oluşturulan yüzey daha esnek bir yapıda olabilmektedir. Temel fonksiyonda x ve y değişkenleri sırasıyla u ve v yönündeki düğümleri, k ve l ise sırasıyla

u ve v yönündeki eğri derecesinin bir fazlasını temsil etmektedir. Şekil 4.10'da bir B-Spline yüzeyi örneği verilmiştir.



Şekil 4.10. B-Spline yüzeyi (Derece 3) [32]

4.2.3. NURBS yüzeyleri

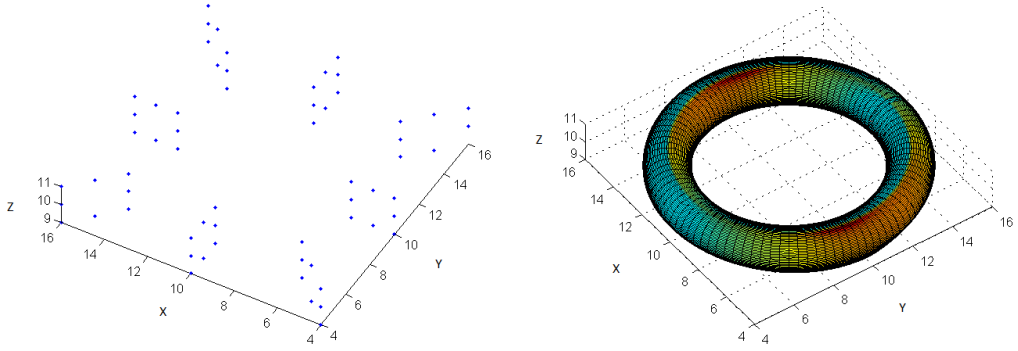
NURBS yüzeyleri bilgisayar grafikleri ve bilgisayarlı tasarım alanında bir standart haline gelmiş yüzey gösterimidir. NURBS yüzeyleri ile birçok karmaşık şekil kolay ve doğru bir şekilde tanımlanıp gösterilebilmektedir. NURBS yüzeyleri rasyonel B-Spline yüzeylerinin özel bir halidir. NURBS, rasyonel B-Spline'ların düğüm vektörlerinin tekdüze olmayan formunun kullanılması ile oluşmuştur [31].

NURBS eğrilerinin iki parametre uzayına genişletilmesiyle oluşmuştur. NURBS eğrilerinin bütün özelliklerini miras almaktadır. Aşağıda Denklem 4.19 ve 4.20'de NURBS yüzeylerinin genel tanımı verilmiştir.

$$Q(u,v) = \frac{\sum_{i=0}^{n}\sum_{j=0}^{m} h_{i,j} B_{i,j} N_{i,k}(u) M_{j,l}(v)}{\sum_{i=0}^{n}\sum_{j=0}^{m} h_{i,j} N_{i,k}(u) M_{j,l}(v)} = \sum_{i=0}^{n}\sum_{j=0}^{m} B_{i,j} S_{i,j}(u,v) \qquad \textbf{(4.19)}$$

$$S_{i,j}(u,v) = \frac{h_{i,j} B_{i,j} N_{i,k}(u) M_{j,l}(v)}{\sum_{i1=0}^{n}\sum_{j1=0}^{m} h_{i1,j1} N_{i1,k}(u) M_{j1,l}(v)} \qquad \textbf{(4.20)}$$

Denklemde $B_{i,j}$ kontrol noktalarını, $N_{i,k}(u)$ ve $M_{j,l}(v)$ B-Spline temel fonksiyonunu, u ve v parametreleri, n ve m sırasıyla u ve v yönündeki kontrol noktaları sayısını, k ve l sırasıyla u ve v yönündeki eğri derecesinin bir fazlasını ve $h_{i,j}$ kontrol noktalarına karşılık gelen ağırlık noktalarını temsil etmektedir. Aşağıda şekil 4.11'de NURBS yüzey örneği verilmiştir.



Şekil 4.11. NURBS yüzeyi(sağ) ve kontrol noktaları(sol) (derece 2)

Şekil 4.11'de sol taraftaki şekilde kontrol noktaları ile oluşturulmuş dış bükey omurga içerisinde sağ tarafta görülen yüzey yer almaktadır. Şekil 4.11'de oluşturulan yüzeyin verileri aşağıdaki Tablo 4.1 ve Tablo 4.2'de verilmiştir. Düğüm vektörleri olarak u ve v yönlerinde [0,0,0,1/4,1/4,2/4,2/4,3/4,3/4,1,1,1] değerleri kullanılmıştır.

Tablo 4.1. Kontrol noktaları (X,Y,Z)

i	$B_{i,0}$	$B_{i,1}$	$B_{i,2}$	$B_{i,3}$	$B_{i,4}$	$B_{i,5}$	$B_{i,6}$	$B_{i,7}$	$B_{i,8}$
0	(15,10,9)	(16,10,9)	(16,10,10)	(16,10,11)	(15,10,11)	(14,10,11)	(14,10,10)	(14,10,9)	(15,10,9)
1	(15,15,9)	(16,16,9)	(16,16,10)	(16,16,11)	(15,15,11)	(14,14,11)	(14,14,10)	(14,14,9)	(15,15,9)
2	(10,15,9)	(10,16,9)	(10,16,10)	(10,16,11)	(10,15,11)	(10,14,11)	(10,14,10)	(10,14,9)	(10,15,9)
3	(5,15,9)	(4,16,9)	(4,16,10)	(4,16,11)	(5,15,11)	(6,14,11)	(6,14,10)	(6,14,9)	(5,15,9)
4	(5,10,11)	(4,10,9)	(4,10,10)	(4,10,11)	(5,10,11)	(6,10,11)	(6,10,10)	(6,10,9)	(5,10,9)
5	(5,5,9)	(4,4,9)	(4,4,10)	(4,4,11)	(5,5,11)	(6,6,11)	(6,6,10)	(6,6,9)	(5,5,9)
6	(10,5,9)	(10,4,9)	(10,4,10)	(10,4,11)	(10,5,11)	(10,6,11)	(10,6,10)	(10,6,9)	(10,5,9)
7	(15,5,9)	(16,4,11)	(16,4,10)	(16,4,11)	(15,5,11)	(14,6,11)	(14,6,10)	(14,6,9)	(15,5,9)
8	(15,10,9)	(16,10,9)	(16,10,10)	(16,10,11)	(15,10,11)	(14,10,11)	(14,10,10)	(14,10,9)	(15,10,9)

Tablo 4.2. Ağırlık değerleri

i	$w_{i,0}$	$w_{i,0}$	$w_{i,0}$	$w_{i,0}$	wi,0	$w_{i,0}$	$w_{i,0}$	$w_{i,0}$	$w_{i,0}$
0	1	1/√2	1	1/√2	1	1/√2	1	1/√2	1
1	1/√2	1/2	1/√2	1/2	1/√2	1/2	1/√2	1/2	1/√2
2	1	1/√2	1	1/√2	1	1/√2	1	1/√2	1
3	1/√2	1/2	1/√2	1/2	1/√2	1/2	1/√2	1/2	1/√2
4	1	1/√2	1	1/√2	1	1/√2	1	1/√2	1
5	1/√2	1/2	1/√2	1/2	1/√2	1/2	1/√2	1/2	1/√2
6	1	1/√2	1	1/√2	1	1/√2	1	1/√2	1
7	1/√2	1/2	1/√2	1/2	1/√2	1/2	1/√2	1/2	1/√2
8	1	1/√2	1	1/√2	1	1/√2	1	1/√2	1

NURBS yüzeylerini oluşturan sözde kod aşağıda verilmiştir, gerçek kod ise Ek'de verilmiştir.

$B_{i,j}$ değerlerini ekle, U düğümlerini ekle, V düğümlerini ekle, W ağırlıklarını belirle, u_span ve v_span değerlerini al, k ve l derecelerini al, $Q_{u,v}$ sonuç nokta oluştur.

```
for u=0:1
{
        v=0;
        for v=0:1
        {
                x=0,y=0,z=0,

                for i=0:n
                        for j=m
                        {
                                x+=B_{i,j}.x*S_{i,j}(U,V,u,v);
                                y+=B_{i,j}.y*S_{i,j}(U,V,u,v);
                                z+=B_{i,j}.z*S_{i,j}(U,V,u,v);
                        }
                Q_{u,v}.x=x;
                Q_{u,v}.y=y;
                Q_{u,v}.z=z;
                v+=v_span;
        }
        u+=u_span;
}
```

Şekil 4.12 NURBS yüzeyi sözde kodu

BÖLÜM 5. YÜZEY GERİ ÇATIM PROBLEMİ

Yüzey geri çatımı problemi verilen bir nokta bulutu üzerinden bir parametrik yüzey elde etme işlemidir. Burada giriş değeri olarak nokta bulutu alınırken çıkış değeri olarak parametrik yüzeyin parametreleri alınmaktadır [38].

Yüzey geri çatımı probleminin temel adımları aşağıdaki şekilde sıralanabilir.

- Nokta bulutunun alınması
- Parametrelerin belirlenmesi
- Sonuçların üretilmesi
- Sonuçların nokta bulutu ile karşılaştırılması ve hata değerinin hesaplanması

5.1. NURBS Yüzeylerinin Geri Çatımı

NURBS yüzeyinin geri çatımı problemini kısaca tanımlamamız gerekirse girdi olarak alınan bir nokta bulutundan, o nokta bulutunu temsil eden NURBS parametreleri olan kontrol noktalarını, u ve v yönlerindeki düğüm noktalarını ve ağırlık noktalarının bulunması işlemidir.

Yukarıda anlatılan işlemi gerçekleştirmek için literatürde şu adımlar uygulanmıştır.

- Eğri derecesi, kontrol noktası sayısı ve düğüm değerleri sabit olarak verilir
- Her bir veri noktasına bir parametre değeri atanır.
- Verilen nokta kümesi ile hesaplanan nokta kümesi arasındaki fark minimize edilir veya sıfırlanır.

Yukarıdaki adımlar arasında en kritik olan adım ikinci adım olan verilere parametre aktarılmasıdır. Bu adımda genellikle iki tür yaklaşım sergilenmektedir. Bunlar Centripetal yöntem ve kiriş uzunluğu yöntemidir. Bunlardan en çok kullanılanı olan Centripetal yöntemin denklemi denklem (5.1)'de verilmiştir.

$$u_0 = 0 \qquad u_m = 1$$

$$u_i = u_{+-1} + \frac{\sqrt{|Q_i - Q_{i-1}|}}{\sum_{j=0}^{m} \sqrt{|Q_i - Q_{i-1}|}} \tag{5.1}$$

Denklemde u düğümü ve Q veri noktasını temsil etmektedir. Düğümler hesaplandıktan sonra aşağıdaki şekilde düğüm vektörü olarak yerleştirilmektedir.

$$U = [0,0,...,0,u_{k+1},...,u_m,1,1,...1]$$
$$j = 1,....,m-k \qquad için \qquad u_{j+k} = \frac{1}{k} \sum_{i=j}^{j+k-1} u_i \tag{5.2}$$

Burada k eğri derecesinin bir fazlasını, U düğüm vektörleri kümesini ve m ise kontrol noktası sayısının k kadar eksiğini temsil etmektedir. Bu yöntem kullanılarak u ve v yönündeki düğümler hesaplanır.

Düğüm noktaları hesaplandıktan sonra bütün ağırlıklar 1 kabul edilip uygun kontrol noktaları nokta bulutuna karşılık gelecek şekilde Denklem 4.19 kullanılarak hesaplanır.

Daha sonra uyuşmayan kontrol noktaları için ağırlıklar üzerinde oynama yapılarak girdi olarak alınan nokta bulutu ile hesaplanan nokta bulutu arasındaki fark minimize edilir. Bu işlemin genel formülü ise aşağıda verilmiştir.

$$E = \left(\sum_{i=0}^{q} |Q_i - S_i| / q \right) \tag{5.3}$$

Burada Q_i girdi olarak alınan nokta bulutunu, S_i hesaplanan nokta bulutunu, q hesaplanan nokta sayısını, E ise hatayı temsil etmektedir.

5.2. NURBS Yüzey Geri Çatımı Probleminin PSO Algoritmasına Uyarlanması

PSO algoritması daha öncede belirtildiği üzere bir optimizasyon algoritmasıdır ve kesin sonuç yerine kesin sonuca yakın bir değer bulmaktadır. Yine optimizasyon algoritmalarının esas amacı olarak bir değeri optimize etmelidir yani minimize veya maksimize etmelidir.

Yüzey geri çatımı problemine geri dönecek olursak bu problemde optimize edilecek amaç fonksiyonu Denklem 5.3'te verilen hata değeridir. Bu denklemin minimize edilmesi ile yaklaşık bir sonuç bulunabilecektir.

Optimize edilecek değişkenler olarak NURBS yüzeylerinin oluşumunda etkili olan kontrol noktaları, düğüm değerleri ve ağırlık değerleri alınacaktır. Bu değişkenlere göre oluşacak yüzey ile Denklem 5.3 yardımıyla girdi olarak alınan yüzey arasındaki hata bulunur.

NURBS yüzeyleri optimize edildiği için ve düğüm değerlerinde açık düğüm modeli kullanılacağından dolayı ilk ve son derece sayısının bir fazlası kadar olan düğüm sayısı hesaplanacak düğümlerden çıkarılacaktır.

Açık düğüm modeli kullanıldığından dolayı yeni oluşan sonuç noktalar, kontrol noktalarının u ve v yönlerindeki son değerleri ile kesiştiğinden dolayı kontrol noktası değişkenlerinin tanım aralığı nokta bulutundaki maksimum ve minimum değerleri olarak belirlenebilir.

Yukarıda anlatılan yöntem özetlenirse aşağıdaki şekilde adımlar oluşturulabilir.

- Eğri derecesi ve kontrol noktası sayısının belirlenmesi
- Nokta bulutunun programa yüklenmesi
- Girilen nokta bulutuna göre minimum ve maksimum değerleri belirlenir
- Rasgele koordinat düğüm ve ağırlık değerleri atanır
- Yüzey oluşturulur ve hata değeri hesaplanır
- Hata değerine göre parçacıklar hareket ettirilir veya program sonlandırılır

BÖLÜM 6. PSO VE ÇS ALGORİTMASININ PROBLEMLER ÜZERİNDEKİ PERFORMANSININ KARŞILAŞTIRILMASI

6.1. Test Fonksiyonları

Test amacıyla kullanılan fonksiyonlar ve tanım aralıkları aşağıdaki Tablo 6.1'de özet olarak verilmiştir. Bu fonksiyonların tercih edilmesinin sebebi birçok çalışmada [13-30] popüler olarak kullanılmasıdır. Kullanılan algoritma parametre değerleri ise özet olarak Tablo 6.2'de verilmiştir.

Tablo 6.1. Test fonksiyonları

Fonksiyon Adı	Tanım Aralığı	Tanımı
Ellipsodial Fonksiyon	(-100, 100)	$f_1(x) = \sum_{i=1}^{n} i\, x_i^2$
Rosenbrock Fonksiyonu	(-30, 30)	$f_2(x) = \sum_{i=1}^{n} (100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2)$
Griewank Fonksiyonu	(-600, 600)	$f_3(x) = \frac{1}{4000}\sum_{i=1}^{n} x_i^2 - \prod_{i=1}^{n} \cos\left(\frac{x_i}{\sqrt{i}}\right) + 1$
Ackley Fonksiyonu	(-32.768, 32.768)	$f_4(x) = 20 + e - 20exp\left(-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2}\right) - exp\left(\frac{1}{n}\sum_{i=1}^{n} \cos(2\pi x_i)\right)$

Tablo 6.2. PSO ve Çoğalan Sürü algoritması parametreleri

Parametreler	PSO	ÇS
Parçacık Sayısı	125	125
C1	1.8	1.8
C2	1.8	1.8
Üreme Oranı	-	0.5
Seçim Şeması	-	Turnuva Seçimi
Turnuva Boyutu	-	3
Mutasyon Oranı	-	1/Boyut

6.2. Test Fonksiyonları Sonuçları

Her bir test fonksiyonunda algoritmalar 10 boyut için 1000 ve 20 boyut için 1500 döngü boyunca çalıştırılmıştır. Elde edilen sonuçların ortalaması ve standart sapması alınmıştır. Sonuçlar özet olarak Tablo 6.3'de verilmiştir.

Tablo 6.3 Test fonksiyonlarına göre PSO ve ÇS algoritmalarının sonuçları

Fonksiyon	Boyut	PSO Ortalama (Standart Sapma)	ÇS Ortalama (Standart Sapma)
Ellipsoidal Fonksiyonu	10	0,125176 (0,39708)	6,33E-08 (-2,26E-07)
	20	6,115002 (9,104639)	0,744034 (0,789478)
Rosenbrock Fonksiyonu	10	4,778433 (2,487662)	2,110912 (2,352057)
	20	13,09686727 (13,8583444)	10,65285311 (2,23640593)
Griewank Fonksiyonu	10	0,004367103 (0,005704754)	0,063425016 (0,098349987)
	20	0,183087907 (0,229020138)	0,197518884 (0,191952133)
Ackley Fonksiyonu	10	1,325490902 (0,914628641)	0,822298525 (0,921644178)
	20	4,34005 (1,172947)	1,755973 (0,736546)

Yukarıdaki sonuçlara göre ÇS algoritması PSO algoritmasından daha iyi sonuçlar elde etmiştir. Boyut sayısı arttıkça ÇS algoritması daha iyi sonuçlar elde ederken düşük boyutlarda ise başa baş sonuçlar elde etmiştir. Elde ettiğimiz sonuçlar literatürde elde edilen sonuçları desteklemektedir [30]. ÇS algoritması birçok kez tam sonuç bulmuş ve yerel minimum noktalarından kaçınmıştır. PSO algoritması ise ÇS algoritmasına göre daha az sayıda tam sonuç bulmuş ve birçok kez yerel minimum noktalarında takılmıştır.

6.3. PSO ve ÇS Algoritmasının Parametreleri

Yapılan çalışmada her iki algoritmada da aynı parametre değerleri ve değişken aralıkları kullanılmıştır. İki adet örnek yüzey üzerinde testler gerçekleştirilmiş olup bu örnek yüzeyler aşağıda Şekil 6.1 ve 6.2'de verilmiştir.

Optimize edilecek değişkenler olarak NURBS yüzey parametreleri olan kontrol noktaları, düğüm değerleri ve ağırlıklar seçilmiştir. Kontrol noktalarının değişken aralığı yüklenen nokta bulutunun minimum ve maksimum değerleri olarak seçilmiştir. Düğüm ve ağırlık değerleri ise [0,1] aralığındadır. NURBS yüzeyleri kullanıldığından düğümler için baştan ve sondan eğri derecesinin bir fazlası kadar veri 0 ve 1 olarak atanmış ve buna göre değişecek olan düğüm sayısı belirlenmiştir.

Parçacık sayısı her iki algoritmada da 60 olarak belirlenmiştir. Sosyal bileşenler olan c1 ve c2 katsayıları 1.8 alınmıştır. Parçacık hızlarına maksimum ve minimum limitler konulmuştur. Topoloji olarak bütün kuşların birbiri ile haberleştiği bir yapı kullanılmıştır. Parçacık hızlarında iç ağırlık olarak kaotik iç ağırlık mekanizması kullanılmıştır.

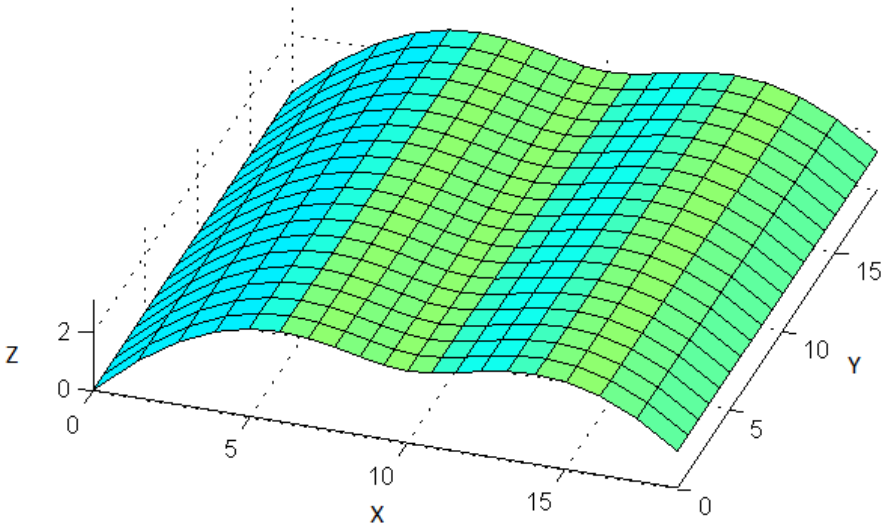
Çoğalan Sürü algoritmasında ise genetik algoritma kısmındaki parametreler olarak üreme oranı 0.5, seçim yöntemi olarak turnuva seçimi, turnuva boyutu olarak 3 ve mutasyon oranı ise 0.01 seçilmiştir. Mutasyon operatörü olarak gerçek kodlu GA üzerinde kullanılan rastgele değerli mutasyon operatörü kullanılmıştır. Özet olarak aşağıdaki Tablo 6.4'de parametre değerleri verilmiştir.

Tablo 6.4. PSO ve Çoğalan Sürü algoritması parametreleri

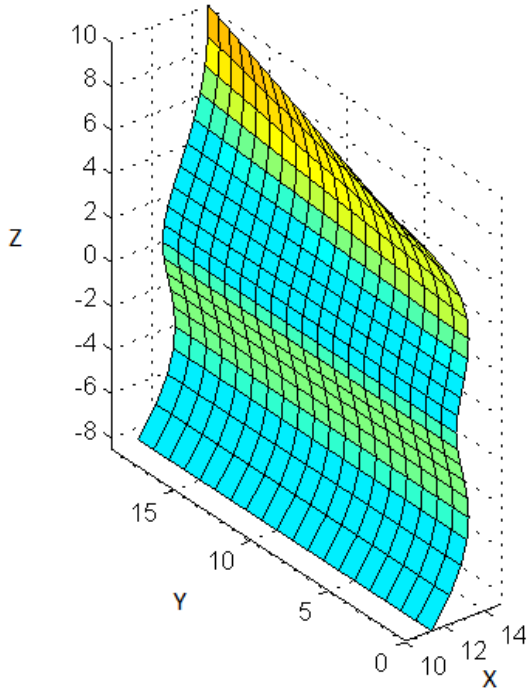
Parametreler	PSO	Çoğalan Sürü
Parçacık Sayısı	60	60
C1	1,8	1,8
C2	1,8	1,8
Üreme Oranı	-	0,5
Seçim Şeması	-	Turnuva Seçimi
Turnuva Boyutu	-	3
Mutasyon Oranı	-	0,01

6.4. Kullanılan Yüzeyler ve Parametreleri

Yapılan çalışmada iki adet yüzey kullanılmıştır. Bu yüzeylerin görselleri Şekil 6.1 ve 6.2'de gösterilmiştir. Parametreleri Tablo 6.5 ve 6.6'te verilmiştir. Her iki yüzeyde 400 adet nokta bulutundan oluşmaktadır. Her iki yüzeyinde bir yönde eğri derecesi 1 diğer yönde ise 3 olarak alınmıştır.



Şekil 6.1. Çalışılan yüzey 1 (Derece u=1 Derece v=3)



Şekil 6.2. Çalışılan yüzey 2 (Derece u=1, Derece v=3)

Tablo 6.5. Kontrol noktaları (X,Y,Z)

i	$B_{i,0}$	$B_{i,1}$
0	(0,0,0)	(0,20,0)
1	(5,0,5)	(5,20,5)
2	(10,0,0)	(10,20,0)
3	(15,0,5)	(15,20,5)
4	(20,0,0)	(20,20,0)

Tablo 6.6. Kontrol noktaları (X,Y,Z)

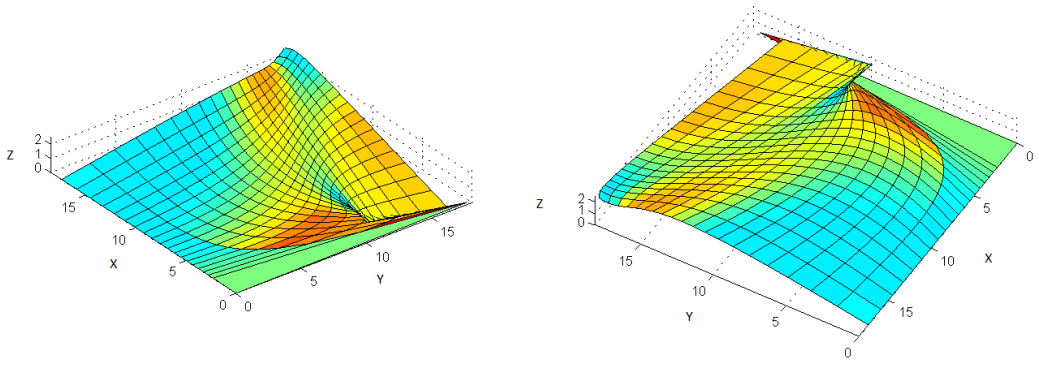
i	$B_{i,0}$	$B_{i,1}$
0	(10,0,10)	(15,20,10)
1	(15,0,5)	(15,20,5)
2	(10,0,0)	(10,20,0)
3	(15,0,-5)	(15,20,-5)
4	(10,0,-10)	(10,20,-10)

Düğüm değerleri Şekil 6.1 ve 6.2 için u yününde [0,0,1,1] ve v yönünde [0,0,0,0,0.5,1,1,1,1] şeklinde girilmiştir.

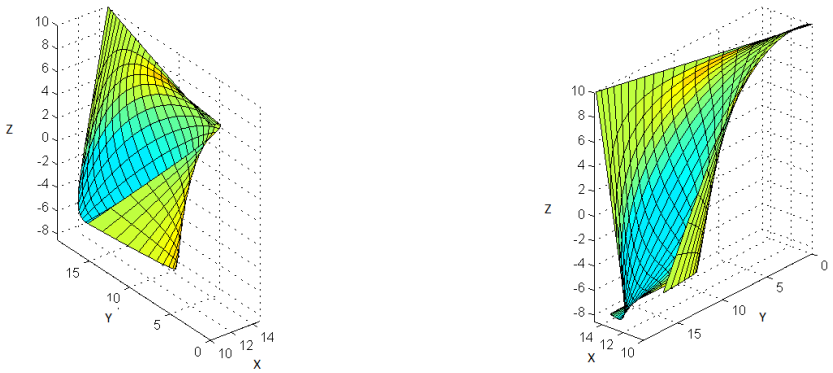
6.5. Sonuçlar

Şekil 6.1 ve 6.2'de gösterilen yüzeyler PSO ve ÇS algoritmaları ile çözülmeye çalışılmış ve yaklaşık sonuç bulunmaya çalışılmıştır. PSO algoritması erken aşamalarda ÇS algoritmasından daha yakın bir sonuç bulmuştur. Her algoritma 1000 döngü boyunca çalıştırılmıştır. Şekil 6.3 ve 6.4'te PSO ile yakınsanan yüzeyler gözükmektedir. Şekil 6.5 ve 6.6'da ise ÇS algoritması ile yakınsanan yüzeyler verilmiştir. PSO ile yakınsanan yüzeylerle ilgili bilgiler şekil 6.3'deki yüzey için Tablo 6.7, 6.8 ve 6.9'de, şekil 6.4'deki yüzey için Tablo 6.10, 6.11 ve 6.12'de verilmiştir. ÇS algoritması ile yakınsanan yüzey ile ilgili bilgiler şekil 6.3'deki yüzey için Tablo 6.13, 6.14 ve 6.15'de, şekil 6.4'deki yüzey için tablo 6.16, 6.17 ve 6.18'de verilmiştir. Her iki yüzey içinde u yönünde derece 2 v yönünde 3 seçilmiştir. U yönündeki kontrol noktası sayısı 3, v yönündeki kontrol noktası sayısı ise 5 seçilmiştir. PSO için 70 döngü, ÇS için 350 döngü süresinde sonuçlar elde edilmiştir. PSO için hata değeri 3,70547837897681330643 olurken ÇS için hata değeri 4,61929571499329556637 olarak bulunmuştur.

Yüzeylere ait hiçbir düğüm ve ağırlık bilgisi programa önceden bildirilmemiş olup girdi olarak sadece nokta bulutunu almıştır ve bu nokta bulutundaki minimum ve maksimum noktaları aralık kabul edilerek rastgele ilk değer ataması gerçekleştirilmiştir. Ağırlık ve düğüm değerleri [0,1] aralığında olduğu için bu bilgilerin ilk ataması yine [0,1] aralığında rastgele olarak atanmıştır.



Şekil 6.3. PSO ile yakınsanan yüzey 1



Şekil 6.4. PSO ile yakınsanan yüzey 2

Tablo 6.7. Kontrol noktaları (X,Y,Z)

i	$B_{i,0}$	$B_{i,1}$	$B_{i,2}$
0	(0,19,0)	(0,19, 1.407)	(0,19,3.125)
1	(18.573,0,0)	(0, 1.874,0)	(0,19,0)
2	(18.573,0,0)	(0,0,3.125)	(18.573,19,3.125)
3	(0,0,3.125)	(18.573,19,0)	(13.339,19,0)
4	(18.573,0,0)	(18.573,19,3.125)	(18.573,19.0)

Tablo 6.8. Ağırlık değerleri

i	$W_{i,0}$	$W_{i,1}$	$W_{i,2}$
0	1	0	1
1	0	1	1
2	1	1	0,830068796742758204510618555 2
3	1	1	0,078986716123720931357174600 8
4	1	1	0

Tablo 6.9. Düğüm değerleri

u	0	0	0	1	1	1	-	-	-
v	0	0	0	0	0	1	1	1	1

Tablo 6.10. Kontrol noktaları (X,Y,Z)

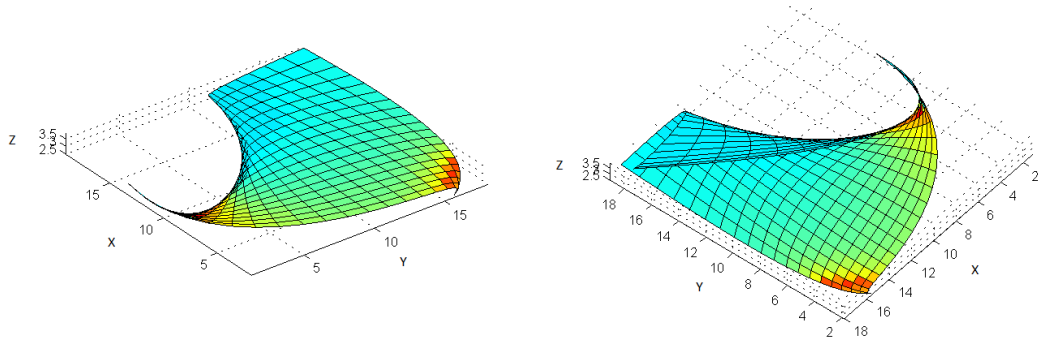
i	$B_{i,0}$	$B_{i,1}$	$B_{i,2}$
0	(10,0,10)	(14.75,19,10)	(14.75,19,10)
1	(11.977,19,10)	(14.75,0,8.182)	(11.267,19,-8.573)
2	(14.75,0,10)	(14.75,16.754,-8.573)	(10,17.974,-8.573)
3	(10,18.499,-8.573)	(14.75,0,-8.573)	(14.75,0,-8.573)
4	(14.75,19,-8.573)	(14.75,0,10)	(14.75,0,-8.573)

Tablo 6.11. Ağırlık değerleri

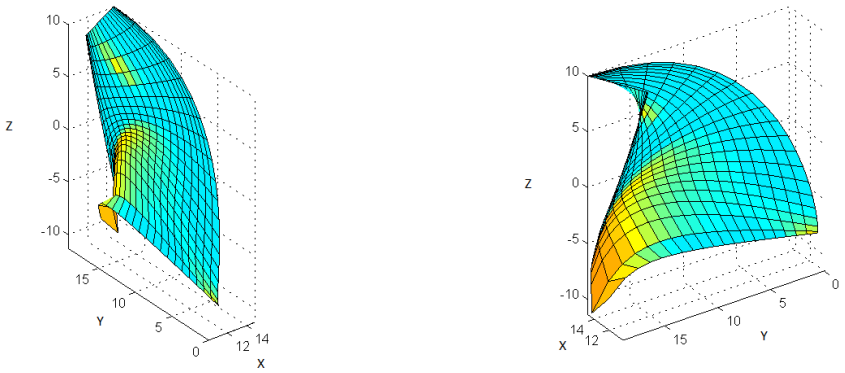
i	$W_{i,0}$	$W_{i,1}$	$W_{i,2}$
0	1	1	0,09965994367500063010855044 39
1	0	0	1
2	1	0,14100129014708718344861459 10	0
3	1	1	0,98814343171845403480858568 09
4	0,93999574647871241166271173 16	0,00570388967896334638042121 72	0

Tablo 6.12. Düğüm değerleri

u	0	0	0	1	1	1	-	-	-
v	0	0	0	0	1	1	1	1	1



Şekil 6.5. ÇS ile yakınsanan yüzey 1



Şekil 6.6. ÇS ile yakınsanan yüzey 2

Tablo 6.13. Kontrol noktaları (X,Y,Z)

i	$B_{i,0}$	$B_{i,1}$	$B_{i,2}$
0	(1.23,13.30,3.125)	(0.272,2.116,1.952)	(18.573,19.3.125)
1	(0.448,1.707,1.080)	(9.675,14.998,3.125)	(5.458,1.880,3.125)
2	(18.573,1.861,3.125)	(6.780,19,1.964)	(6.577,19,2.076)
3	(14.475,1.790,2.041)	(18.573,2.317,3.125)	(18.573,19,3.125)
4	(0.878,1.909,3.125)	(18.573,1.948,2.043)	(1.033,19,3.125)

Tablo 6.14. Ağırlık değerleri

i	$W_{i,0}$	$W_{i,1}$	$W_{i,2}$
0	1	1	1
1	0,907507815879779324739530 4907	1	0
2	1	0,214388491878035504923940 4791	1
3	0,411597255233708507389312 6634	1	1
4	1	1	1

Tablo 6.15. Düğüm değerleri

u	0	0	0	1	1	1	-	-	-
v	0	0	0	0	1	1	1	1	1

Tablo 6.16. Kontrol noktaları (X,Y,Z)

i	$B_{i,0}$	$B_{i,1}$	$B_{i,2}$
0	(14.75,19,10)	(14.75,19,-8.573)	(10,16.640,10)
1	(14.75,0,10)	(10,19,-8.573)	(14.75,19,-8.573)
2	(10,0,-8.573)	(14.75,19,10)	(14.75,19,-8,573)
3	(11.179,0,-8.573)	(10,19,-8.573)	(10,19,10)
4	(10,19,-8.573)	(10,19,10)	(10,19,10)

Tablo 6.17. Ağırlık değerleri

i	$W_{i,0}$	$W_{i,1}$	$W_{i,2}$
0	1	0	1
1	1	0,20673755368330946868559003 29	0,37055453295018505204843492 47
2	1	1	0,54172818622267264028904907 04
3	1	0,42585805077875641915372515 32	0
4	0	1	0,69092345987370466031149290 90

Tablo 6.18. Düğüm değerleri

u	0	0	0	1	1	1	-	-	-
v	0	0	0	0	0,922	1	1	1	1

ÇS algoritmasında PSO algoritmasına göre döngü sayısı açısından bakıldığında daha geç ve daha kötü bir yakınsama gerçekleşmiştir. Bunun sebebi iyi olarak yakınsayan parçacıkların sürünün yok edilen kısmına denk gelebilmelerinden kaynaklanmaktadır. Bu parçacıklar yok edildiğinden dolayı algoritma daha fazla yerel arama yapamamıştır ve bu sebeple bulduğu küresel çözümü geliştirememiştir.

Bütün yazılımlar C# dilinde Visual Studio 2012 kullanılarak .Net 4.5 ortamında yazılmıştır. Oluşturulan yüzeylerin çizimleri için Matlab programı kullanılmıştır. Bütün testler 3.20Ghz hızına sahip Intel Core i5 işlemcili 8Gb Ram belleğe sahip bilgisayar üzerinde gerçekleştirilmiştir.

BÖLÜM 7. SONUÇLAR VE ÖNERİLER

Sonuç olarak her ne kadar ÇS algoritması test fonksiyonları üzerinde iyi sonuç sergilemiş olsa da ele alınan problemin çözümü için kötü sonuç sergilemiştir. Bu sebeple ÇS algoritması genel yapısı ve işleyiş mekanizmasından dolayı bu problemin çözümü için uygun değildir. PSO ise genel olarak belirli bir süre sonra sadece belirli bölgelerde yerel arama yapıp o bölgelere takıldığından dolayı çalışma süresinin sonlarına doğru küresel en iyi sonucu geliştirememiştir.

Bu tez çalışmasında iki algoritmanın karşılaştırılması amaçlandığından dolayı problem mümkün olduğu kadar zor bir hale getirilmiştir. Tam çözümü elde etmek için yüzey geri çatımı probleminde optimize edilecek olan değişkenlerin sayısının azaltılması yoluna gidilebilir. Mevcut problemde kontrol noktaları rastgele atandığından dolayı karmaşık ve girdiye benzemeyen şekiller elde edilmiştir. Benzerliğin artırılması için kontrol noktaları sabit olarak belirli bir kurala göre eklenilebilir. Bu sayede problem çözülmesi kolay bir hale getirilmiş olur.

PSO ve ÇS ile yaptığımız çalışmada bu algoritmaların birçok parametresi sonucu doğrudan ya da dolaylı olarak etkilediğini görülmüştür. Örnek olarak algoritmalarda hız kısıtlamaları uygulandığında algoritma tek bir noktaya kilitlenerek hep bir yönde ilerleme yaptığından dolayı belirli bir süre sonra parçacıklar eksenin maksimum ve minimum noktalarında yığılmışlar ve ilerleme göstermemişlerdir. Ancak hız kısıtlaması uygulanmadığında yaklaşılan yüzey hatası azalmıştır. Parçacıklar döngü sınırına ulaşana kadar arama uzayının çoğunu aramışlardır.

Her iki algoritmanın da parametreleri değiştirilerek bu parametrelerin çözüm üzerindeki etkileri araştırılabilir. Özellikle ÇS algoritmasında genetik algoritma seçim şeması, çaprazlama yöntemi ve mutasyon yöntemi ile birey yok etme mekanizması değiştirilebilir. Bu değişikliklerin çözüm üzerindeki etkileri araştırılabilir

Bu tez çalışmasında her bir parçacığın birbirine bağlı olduğu gbest topoloji yapısı kullanıldığından parçacıklar her döngüde farklı bir parçacığa doğru gitme eğilimi göstermiştir. Ancak kat edebildikleri yol kısıtlı olduğundan küme olarak arama yapamamışlardır. Bu

sebeple bu problemin çözümünde farklı topolojiler kullanılabilir ve bu topolojilerin çözüm üzerine etkisi incelenebilir.

KAYNAKLAR

[1] WEINERT, K., MEHNEN, J., DRERUP, P., ALBERSMANN, F., New solutions for surface reconstruction from discrete point data by means of computational intelligence, Universität Dortmund, 1998. http://hdl.handle.net/2003/5350.

[2] WEINERT, K., SURMANN, T., MEHNEN, J., Evolutionary surface reconstruction using CSG-NURBS hybrids. Universität Dortmund, 2001. http://hdl.handle.net/2003/5413.

[3] WAGNER, T., MICHELITSCH, T., SACHAROW, A., On the design of optimisers for surface reconstruction. ACM conference on Genetic and evolutionary computation (GECCO'07) , pp. 2195-2202, 2007.

[4] SEN, S., ZHENG, S., Near-optimal triangulation of a point set by simulated annealing. ACM symposium on Applied computing: technological challenges of the 1990's, pp. 1000-1008, 1992.

[5] GOINSKI, A., Evolutionary surface reconstruction. IEEE Conference on Human System Interactions, Krakow, pp. 464-469, 2008.

[6] KODAMA, T., LI, X., NAKAHIRA, K., ITO, D., Evolutionary computation applied to the reconstruction of 3-D surface topography in the SEM. Journal of Electron Microscopy, Volume 54:5, pp. 429-435, doi: 10.1093/jmicro/dfi062, 2005.

[7] GÁLVEZ, A., IGLESIAS, A., Particle swarm optimization for non-uniform rational B-spline surface reconstruction from clouds of 3D data points. Information Sciences, Volume 192, pp. 174-192, 2012.

[8] HOLLAND, J.H., Adaptation in natural and artificial systems. Ann Arbor: The University of Michigan Press, 1975.

[9] DORIGO, M., BIRATTARI, M., STUTZLE, T., Ant colony optimization. Computational Intelligence Magazine, IEEE (Volume:1 , Issue: 4),pp. 28-39, 2006.

[10] KARABOĞA, D., BAŞTÜRK, B., A powerful and efficient algorithm for numerical function optimization: artificial bee colony (ABC) algorithm. Journal of Global Optimization, Volume 39, Issue 3, US, 459-471, 2007.

[11] KARABOĞA, D., Yapay Zeka Optimizasyon Algoritmaları. Nobel Yayın Dağıtım, 1 - 18, pp. 182 – 196, 2011.

[12] KENNEDY, J., EBERHART, R.C., , Particle swarm optimization. IEEE International Conference on Neural Networks, Piscataway, NJ, pp. 1942–1948, 1995.

[13] EBERHART, R.C., SHI, Y., Comparing inertia weights and constriction factors in particle swarm optimization. IEEE congress evolutionary computation, San Diego, pp. 84–88, 2000.

[14] SHI, Y., EBERHART, R., A modified particle swarm optimizer. IEEE World Congress on Computational Intelligence, NJ, pp. 69–73, 1998.

[15] BANSAL, J.C., SINGH, P.K., SARASWAT, M., VERMA, A., JADON, S.S., ABRAHAM, A., Inertia weight strategies in particle swarm optimization. Nature and Biologically Inspired Computing (NaBIC), Salamanca, pp. 633-640, 2011.

[16] XIN, J.,CHEN G., HAI, Y., a particle swarm optimizer with multi-stage linearly-decreasing inertia weight. IEEE Computational Sciences and Optimization, Sanya, Hainan, pp. 505-508, 2009.

[17] EBERHART, R.C., SHI, Y., Tracking and optimizing dynamic systems with particle Sürü. Evolutionary Computation, IEEE Proceedings of the 2001 Congress on (Volume:1), Seoul, pp. 94-100, 2001.

[18] CHEN, G., HUANG, X., JIA, J, MIN, Z., Natural exponential inertia weight strategy in particle swarm optimization. IEEE Intelligent Control and Automation, Dalian, pp. 3672-3675, 2006.

[19] LI, H.R., GAO, Y.L., Particle swarm optimization algorithm with exponent decreasing inertia weight and stochastic mutation. IEEE Information and Computing, Manchester, pp. 66-69, 2009.

[20] FENG, Y., TENG, G.F., WANG, A.X., YAO, Y.M., Chaotic inertia weight in particle swarm optimization. IEEE Innovative Computing. Information and Control, Kumamoto, pp. 475, 2007.

[21] ARUMUGAM, M.S., RAO, M.V.C., On the performance of the particle swarm optimization algorithm with various inertia weight variants for computing optimal control of a class of hybrid systems. Discrete Dynamics in Nature and Society, NY, Makale ID:79295, 2006.

[22] EBERHART, R.C., SIMPSON, P., DOBBINS, R., Computational intelligence PC tools. AP Professional, San Diego, CA, Bölüm 6, pp. 212–226, 1996.

[23] KENNEDY, J., Small worlds and mega-minds: effects of neighborhood topology on particle swarm performance. IEEE Evolutionary Computation, Washington DC, Bölüm 3, pp. 1931-1938, 1999.

[24] ANGELINE, P.J., Using selection to improve particle swarm optimization. IEEE evolutionary computation, Anchorage, AK, pp. 84-89, 1998.

[25] BRITS, R., ENGELBRECHT, A.P., VAN DEN BERGH, F., , A niching particle swarm optimizer. Fourth Asia-Pacific conference on simulated evolution and learning, 2002.

[26] VAN DEN BERGH, F., ENGELBRECHT, A.P., A new locally convergent particle swarm optimiser. IEEE Systems, Man and Cybernetics, Bölüm 3, 2002.

[27] KENNEDY, J., The particle swarm: social adaptation of knowledge. Proceedings of international conference evolutionary computation, IEEE, Piscataway, NJ , pp 303–308, 1997.

[28] KRINK, T., LØVEBJERG, M., The lifecycle model: combining particle swarm optimization genetic algorithms and hillclimbers. Parallel Problem Solving from Nature — PPSN VII, Berlin, pp. 621-630, 2002.

[29] HIGASHI, N., IBA, H., Particle swarm optimization with Gaussian mutation. IEEE swarm intelligence symposium (SIS), Indianapolis, Indiana, USA, pp. 72-79, 2003.

[30] SETTLES, M., SOULE, T., Breeding swarms: a GA/PSO hybrid. ACM conference on Genetic and evolutionary computation (GECCO '05), NY, pp. 161-168, 2005.

[31] ROGERS, D.F., An introduction to NURBS with historical perspective. Academic Press, 2001.

[32] PIEGL, L., TILLER, W., The NURBS Book 2nd Edition. Springer-Verlag, 1997.

[33] SCHOENBERG, I.J., Contributions to the problem of approximation of equidistant data by analytic functions. I. J. Schoenberg Selected Papers, Birkhäuser Boston, pp. 3-57, 1988.

[34] COX, M.G., The numerical evaluation of B-splines. IMA Journal of Applied Mathematics, Volume 10:2, pp. 134-149, 1972.

[35] BOOR, C., On calculating with B-splines. Journal of Approximation Theory, Volume 6:1, pp. 50-62, 1972.

[36] RIESENFELD, R.F., Application of B-spline Approximation to Geometric Problems of Computer Aided Design. Syracuse University Doktora Tezi, 1973.

[37] GORDON, W.J., RIESENFELD, R.F., Bernstein-Bézier methods for the computer-aided design of free-form curves and surfaces. Journal of the ACM, Volume 21:2, pp. 293-310, 1974.

[38] ÜLKER, E., Yapay zeka teknikleri kullanılarak yüzey modelleme. Selçuk Üniversitesi Doktora Tezi, 2007.

EK

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Diagnostics;
using System.Drawing;
using System.IO;
using System.Linq;
using System.Reflection;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using Excel = Microsoft.Office.Interop.Excel;
namespace NURBS_Surface
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }
        public class Koordinat
        {
           public decimal x;
           public decimal y;
           public decimal z;
        }
        DataTable kd = new DataTable();
        string path = "";
        int derece_u;
        int derece_v;
        int Dugum_u_sayisi;
        int Dugum_v_sayisi;
        int kontrol_nok_u;
        int kontrol_nok_v;
        int agirlik_sayisi;
        decimal[,] agirliklar;
        Koordinat[,] kontrol_noktasi_agi;
        Koordinat[,] Sonuçlar;
        decimal u = 0;
        decimal v = 0;
        decimal adım = 1 / 20M;
        private void Form1_Load(object sender, EventArgs e)
        {
            kd.Columns.Add("Nokta Adı");
            kd.Columns.Add("X");
            kd.Columns.Add("Y");
            kd.Columns.Add("Z");
            derece_u = ((int)numericUpDown1.Value);
            derece_v = ((int)numericUpDown2.Value);
```

```
            kontrol_nok_u = (int)numericUpDown3.Value;
            kontrol_nok_v = (int)numericUpDown4.Value;
            Dugum_u_sayisi = listBox1.Items.Count;
            Dugum_v_sayisi = listBox2.Items.Count;
            agirlik_sayisi = listBox3.Items.Count;
        }

        public decimal Nik(int i, int k, decimal u, decimal[] t)
        {
            decimal part1, part2, part3, part4, part5, part6, part7, part8,
part9, part10, part11;
            if (k == 1)
            {
                if ((t[i] <= u) && (u <= t[i + 1]))
                {
                    return 1;
                }
                else
                    return 0;
            }
            else
            {
                part1 = u - t[i];
                part2 = t[i + (k - 1)] - t[i];
                part3 = Nik(i, (k - 1), u, t);
                part4 = t[i + k] - u;
                part5 = t[i + k] - t[i + 1];
                part6 = Nik((i + 1), (k - 1), u, t);
                if (part2 == 0)
                    part7 = 0 / 1;
                else
                    part7 = part1 / part2;
                part8 = part7 * part3;
                if (part5 == 0)
                    part9 = 0 / 1;
                else
                    part9 = part4 / part5;
                part10 = part9 * part6;
                part11 = part8 + part10;
                return Convert.ToDecimal(part11);
            }

        }
        decimal rij_func(int i, int dereceu, decimal u, decimal[] dugumdiziu,
int j, int derecev, decimal v, decimal[] dugumdiziv,decimal [,]agirliklar)
        {
            decimal part1 = Nik(i, dereceu, u, dugumdiziu);
            decimal part2 = Nik(j, derecev, v, dugumdiziv);
            decimal part3 = agirliklar[i,j];
            return 0;
        }
        private void button1_Click(object sender, EventArgs e)
        {
            FileDialog file = new OpenFileDialog();
            file.Filter = "Excel 2007-2010 (*.xlsx)|*.xlsx|Excel 2003
(*.xls)|*.xls|Tüm Dosyalar (*.*)|*.*";
            if (file.ShowDialog() == DialogResult.OK)
            {
                path = file.FileName;
            }
            if (path == "")
```

```
            {
                MessageBox.Show("Dosya Seçilmedi");
                return;
            }
            Excel.Application excelApp = new Excel.Application();
            Excel.Workbook workbook;
            Excel.Worksheet worksheet;
            Excel.Range range;
            workbook = excelApp.Workbooks.Open(path);
            worksheet = (Excel.Worksheet)workbook.Sheets["Veri"];
            int column = 0;
            int row = 0;
            range = worksheet.UsedRange;
            for (row = 2; row <= range.Rows.Count; row++)
            {
                DataRow dr = kd.NewRow();
                for (column = 1; column <= range.Columns.Count; column++)
                {
                    dr[column - 1] = (range.Cells[row, column] as
Excel.Range).Value2.ToString();
                }
                kd.Rows.Add(dr);
                kd.AcceptChanges();
            }
            workbook.Close(true, Missing.Value, Missing.Value);
            excelApp.Quit();
            dataGridView1.DataSource = kd;
            derece_u = ((int)numericUpDown1.Value);
            derece_v = ((int)numericUpDown2.Value);
            kontrol_nok_u = (int)numericUpDown3.Value;
            kontrol_nok_v = (int)numericUpDown4.Value;
            Dugum_u_sayisi = listBox1.Items.Count;
            Dugum_v_sayisi = listBox2.Items.Count;
            agirlik_sayisi = listBox3.Items.Count;
            label8.Text = "Eklenecek Düğüm Sayısı (u) = " + (kontrol_nok_u +
derece_u + 1 - Dugum_u_sayisi).ToString();
            label9.Text = "Eklenecek Düğüm Sayısı (v) = " + (kontrol_nok_v +
derece_v + 1 - Dugum_v_sayisi).ToString();
            label12.Text = "Eklenecek Ağırlık Sayısı = " + ((kontrol_nok_u *
kontrol_nok_v) - agirlik_sayisi).ToString();
        }

        StreamWriter tw;
        private void button2_Click(object sender, EventArgs e)
        {

            Sonuçlar = new Koordinat[20, 20];
            kontrol_noktasi_agi = new Koordinat[kontrol_nok_v, kontrol_nok_u];
            agirliklar = new decimal[kontrol_nok_v, kontrol_nok_u];
            for (int i = 0; i < 20; i++)
            {
                for (int j = 0; j < 20; j++)
                {
                    Sonuçlar[i, j] = new Koordinat();
                }
            }
            for (int i = 0; i < kontrol_nok_v; i++)
            {
                for (int j = 0; j < kontrol_nok_u; j++)
                {
                    kontrol_noktasi_agi[i, j] = new Koordinat();
```

```
                }
            }
            decimal [] dugum_dizi_u=new decimal[Dugum_u_sayisi];
            decimal [] dugum_dizi_v=new decimal[Dugum_v_sayisi];


            int sayac = 0;
            for (int i = 0; i < kontrol_nok_v; i++)
            {
                for (int j = 0; j < kontrol_nok_u; j++)
                {
                    agirliklar[i, j]
=Convert.ToDecimal(listBox3.Items[sayac]);
                    sayac++;
                }
            }
            sayac = 0;
            for (int i = 0; i < kontrol_nok_v; i++)
            {
                for (int j = 0; j < kontrol_nok_u; j++)
                {
                    kontrol_noktasi_agi[i, j].x =
Convert.ToDecimal(kd.Rows[sayac][1]);
                    kontrol_noktasi_agi[i, j].y =
Convert.ToDecimal(kd.Rows[sayac][2]);
                    kontrol_noktasi_agi[i, j].z =
Convert.ToDecimal(kd.Rows[sayac][3]);
                    sayac++;
                }
            }
            for(int i=0;i<listBox1.Items.Count;i++)
            {
                dugum_dizi_u[i]=Convert.ToDecimal(listBox1.Items[i]);
            }
            for(int i=0;i<listBox2.Items.Count;i++)
            {
                dugum_dizi_v[i]=Convert.ToDecimal(listBox2.Items[i]);
            }

            Stopwatch sw=new Stopwatch();
            sw.Start();
            //ana fonksiyon
            decimal bolunen_kisim = 0;
            decimal xsonuc=0;
            decimal ysonuc=0;
            decimal zsonuc=0;
            decimal p1=0, p2=0, p3=0, p4x=0, p4y=0, p4z=0, p5=0;
            tw = new StreamWriter("D:\\nurbs_yuzey.txt");
            for (int donecekv = 0; donecekv <20; donecekv++) //donen v kısımı
            {
                u = 0;
                for (int doneceku = 0; doneceku < 20; doneceku++) //donen u
kısımı
                {
                    xsonuc = 0;
                    ysonuc = 0;
                    zsonuc = 0;
                    bolunen_kisim = 0;
                    for (int k = 0; k < kontrol_nok_v; k++)//Rij deki bölünen
sabit kısım burası değişken kısım sadece u ve v
                    {
```

```
                        for (int l = 0; l < kontrol_nok_u; l++)
                        {
                            decimal part1 = Nik(k, (derece_v + 1), v,
dugum_dizi_v);
                            decimal part2 = Nik(l, (derece_u + 1), u,
dugum_dizi_u);
                            decimal part3 = agirliklar[k, l];
                            bolunen_kisim = bolunen_kisim + (part1 * part2 *
part3);
                        }
                    }
                    for (int i = 0; i < kontrol_nok_v; i++)
                    {
                        for (int j = 0; j < kontrol_nok_u; j++)
                        {

                            p1 = Nik(i, (derece_v + 1), v, dugum_dizi_v);
                            p2 = Nik(j, (derece_u + 1), u, dugum_dizi_u);
                            p3 = agirliklar[i, j];
                            p4x = kontrol_noktasi_agi[i, j].x;
                            p4y = kontrol_noktasi_agi[i, j].y;
                            p4z = kontrol_noktasi_agi[i, j].z;
                            if (bolunen_kisim == 0)
                                p5 = 0;
                            else
                            p5 = (p1 * p2 * p3)/bolunen_kisim;
                            xsonuc = xsonuc + (p4x * p5);
                            ysonuc = ysonuc + (p4y * p5);
                            zsonuc = zsonuc + (p4z * p5);
                        }
                    }

                    tw.WriteLine("x=" + xsonuc.ToString() + "\ty=" +
ysonuc.ToString() + "\tz=" + zsonuc.ToString());
                    Sonuçlar[donecekv, doneceku].x = xsonuc;
                    Sonuçlar[donecekv, doneceku].y = ysonuc;
                    Sonuçlar[donecekv, doneceku].z = zsonuc;
                    u = u + adım;
                }
                v = v + adım;
            }
            tw.Close();
            sw.Stop();
            MessageBox.Show("İşlem Tamam " + sw.ElapsedMilliseconds.ToString()
+ "MSn");
        }
```

ÖZGEÇMİŞ

Hüseyin Demirci, 18.04.1990 tarihinde Konya'da doğdu. İlköğretim ve ortaöğretim eğitimini Konya'da muhtelif okullarda tamamladı. 2007 yılında başladığı Konya Selçuk Üniversitesi Bilgisayar Sistemleri Eğitimi Bölümü'nden 2012 yılında mezun oldu. 2012 yılında Konya Selçuk Üniversitesi Fen Bilimleri Enstitüsü Bilgisayar Mühendisliği Anabilim Dalı'nda yüksek lisans eğitimine başladı. 2014 yılında Sakarya Üniversitesi Fen Bilimleri Enstitüsü Bilgisayar ve Bilişim Mühendisliği Anabilim Dalı'nda yüksek lisans eğitimine yatay geçiş ile geçti. 2013 yılından beri Sakarya Üniversitesi Teknoloji Fakültesi Bilgisayar Mühendisliği bölümünde araştırma görevlisi olarak çalışmaktadır.