**YAPAY ZEKÂ TEKNİKLERİ**

**GENETİK ALGORİTMA**

**PARÇACIK SÜRÜ OPTİMİZASYONU**

**ANTEN DİZİLERİNİN OPTİMİZASYONU**

**UYGULAMA HAKKINDA**

**OPTİMİZASYON SONUÇLARI**

**MART 2023**

**ANKARA**

# İÇİNDEKİLER

[İÇİNDEKİLER v](#_Toc130759296)

[ŞEKİLLERİN LİSTESİ vii](#_Toc130759297)

[RESİMLERİN LİSTESİ vii](#_Toc130759298)

[SİMGELER VE KISALTMALAR vii](#_Toc130759299)

[1. GİRİŞ 1](#_Toc130759300)

[2. META-SEZGİSEL ALGORİTMALAR 2](#_Toc130759301)

[2.1 Genetik Algoritma (GA) 2](#_Toc130759302)

[2.2 Parçacık Sürü Optimizasyonu (PSO) 4](#_Toc130759303)

[3. ANTEN DİZİLERİNİN OPTİMİZASYONU 5](#_Toc130759304)

[3.1 Genetik Algoritma ile Anten Dizilerinin Optimizasyonu 6](#_Toc130759305)

[3.1.1 Algoritma 6](#_Toc130759306)

[3.1.2 Değişkenlerin tanımlanması 7](#_Toc130759307)

[3.1.3 Bireyler ve bireylerin kimlik bilgileri 7](#_Toc130759308)

[3.1.4 Kod çözücü 7](#_Toc130759309)

[3.1.5 Dizi faktörü 7](#_Toc130759310)

[3.1.6 Gen üretici 8](#_Toc130759311)

[3.1.7 Kromozom üretici 8](#_Toc130759312)

[3.1.8 Uygunluk değerinin hesaplanması 8](#_Toc130759313)

[3.1.9 Seleksiyon işlemi 8](#_Toc130759314)

[3.1.10 Çaprazlama işlemi 9](#_Toc130759315)

[3.1.11 Mutasyon işlemi 10](#_Toc130759316)

[3.1.12 Genetik algoritma 10](#_Toc130759317)

[3.2 PSO ile Anten Dizilerinin Optimizasyonu 10](#_Toc130759318)

[3.2.1 Algoritma 10](#_Toc130759319)

[3.2.2 Değişkenlerin tanımlanması 11](#_Toc130759320)

[3.2.3 Parçacıklar ve parçacıkların kimlik bilgileri 12](#_Toc130759321)

[3.2.4 Dizi faktörü 12](#_Toc130759322)

[3.2.5 Parçacık üretici 12](#_Toc130759323)

[3.2.6 Parçacıkların hız ve konum bilgilerinin güncellenmesi 12](#_Toc130759324)

[3.2.7 Uygunluk değerinin hesaplanması 12](#_Toc130759325)

[3.2.8 Parçacık sürü optimizasyonu 12](#_Toc130759326)

[4. UYGULAMA HAKKINDA 13](#_Toc130759327)

[4.1 Arayüzün Genel Yapısı 13](#_Toc130759328)

[4.2 Antenler Hakkında Kaynak 13](#_Toc130759329)

[4.3 Yapay Zekâ Hakkında Kaynak 14](#_Toc130759330)

[4.4 Optimizasyon İşlemlerinin Gerçekleştirilmesi 15](#_Toc130759331)

[4.4.1 Genetik algoritma ile optimizasyon işlemleri 16](#_Toc130759332)

[4.4.2 Parçacık sürü optimizasyonu ile optimizasyon işlemleri 17](#_Toc130759333)

[4.5 Grafik Karşılaştırma İşlemleri 18](#_Toc130759334)

[4.6 Programın Çalışabilmesi İçin Gerekli Ayarlamaların Yapılması 19](#_Toc130759335)

[4.7 Proje Hakkında Bilgiler 20](#_Toc130759336)

[4.8 Veri Tabanı 21](#_Toc130759337)

[5. OPTİMİZASYON SONUÇLARI 24](#_Toc130759338)

# ŞEKİLLERİN LİSTESİ

**Şekil Sayfa**

[Şekil 3.1. Genetik algoritmanın akış diyagramı 6](#_Toc130759339)

[Şekil 3.2. Rulet tekerleği yöntemi 8](#_Toc130759340)

[Şekil 3.3. Noktasal çaprazlama yöntemi 10](#_Toc130759341)

[Şekil 3.4. Parçacık sürü optimizasyonunun akış diyagramı 11](#_Toc130759342)

# RESİMLERİN LİSTESİ

**Resim Sayfa**

[Resim 3.1. Farklı Anten sayılarında dizi faktörü 5](#_Toc130759343)

[Resim 4.1. Arayüzün ana sayfası 13](#_Toc130759344)

[Resim 4.2. Anten sayfası 14](#_Toc130759345)

[Resim 4.3. Yapay zekâ sayfası 15](#_Toc130759346)

[Resim 4.4. Optimizasyon sayfası 16](#_Toc130759347)

[Resim 4.5. Genetik algoritma paneli 17](#_Toc130759348)

[Resim 4.6. Parçacık sürü optimizasyonu paneli 18](#_Toc130759349)

[Resim 4.7. Grafikler paneli 19](#_Toc130759350)

[Resim 4.8. Ayarlar sayfası 20](#_Toc130759351)

[Resim 4.9. Hakkında sayfası görünüm 1 21](#_Toc130759352)

[Resim 4.10. Hakkında sayfası görünüm 2 21](#_Toc130759353)

[Resim 4.11. Veri tabanında veriler 22](#_Toc130759354)

[Resim 4.12 Veri tabanında görselleştirilmiş veriler 23](#_Toc130759355)

[Resim 5.1. Örnek optimizasyon sonucu 1 24](#_Toc130759356)

[Resim 5.2. Örnek optimizasyon sonucu 2 25](#_Toc130759357)

# SİMGELER VE KISALTMALAR

|  |  |
| --- | --- |
| **Kısaltmalar** | **Açıklama** |
| GA | Genetic Algorithm |
| PSO | Particle Swarm Optimization |
| AF | Array Factor |
| AI | Artificial Intelligence |
| WPF | Windows Presentation Foundation |
| XAML | Extensible Application Markup Language |

# GİRİŞ

Günümüzde, yapay zekâ ve sezgisel algoritmalar bilgisayar bilimleri ve endüstriyel uygulamalar alanında hızla büyüyen bir disiplindir. Yapay zekâ, bilgisayarların akıllıca davranmalarına ve karmaşık problemleri çözmelerine olanak tanıyan bir teknoloji olarak tanımlanır. Bu teknoloji, otomasyon, robotik, görüntü işleme, doğal dil işleme, ses işleme, öğrenme ve diğer birçok uygulama alanında kullanılır. Sezgisel algoritmalar, doğadaki optimize edilmiş süreçlerden esinlenerek, karmaşık problemleri çözmek için kullanılan matematiksel ve istatistiksel tekniklerdir. Bu algoritmaların temel amacı, mümkün olan en iyi sonucu elde etmek için verileri analiz etmek ve işlemektedir. Yapay zekâ ve sezgisel algoritmaların kullanımı, birçok endüstriyel uygulamada iş süreçlerini optimize etmek, müşteri deneyimini geliştirmek, verimliliği arttırmak ve karlılığı arttırmak için yaygın olarak kullanılmaktadır. Örneğin, müşteri hizmetleri ve destek süreçleri için doğal dil işleme algoritmaları, robotik sistemler için hareket planlama algoritmaları, veri madenciliği ve iş zekâsı için sezgisel algoritmalar kullanılmaktadır. Bu dokümanda, sezgisel algoritmaların temellerini, farklı türlerini ve uygulama alanlarından biri olan, anten dizilerinin optimizasyonunu ele alacağız.

# META-SEZGİSEL ALGORİTMALAR

Meta-sezgisel algoritmalardan kısaca bahsedecek olursak; Bilgisayar bilimlerinde, sezgisel ya da buluşsal bir problem çözme tekniğidir. Sonucun doğruluğunun kanıtlanabilir olup olmadığını önemsememektedir fakat genelde iyiye yakın çözüm yolları elde eder. Sezgisel algoritmalar geçiş süresinde daha verimli hale gelebilmek için en iyi çözümü aramaktan vaz geçerek çözüm zamanını azaltan algoritmalardır. Sezgisel algoritmalar en iyi sonucu bulacaklarını garanti etmezler fakat makul bir süre içerisinde bir çözüm elde edeceklerini garanti ederler. Genellikle en iyiye yakın olan çözüm yoluna hızlı ve kolay bir şekilde ulaşırlar. Sezgisel optimizasyon algoritmaları ise; Büyük boyutlu optimizasyon problemleri için, kabul edilebilir sürede optimuma yakın çözümler verebilen algoritmalardır. Genel amaçlı sezgisel optimizasyon algoritmaları, biyoloji tabanlı, fizik tabanlı, sürü tabanlı, sosyal tabanlı, müzik tabanlı ve kimya tabanlı olmak üzere altı farklı grupta değerlendirilmektedir. Sürü zekâsı tabanlı optimizasyon algoritmaları kuş, balık, kedi ve arı gibi canlı sürülerinin hareketlerinin incelenmesiyle geliştirilmiştir.

Biz bu lisans tezimizde literatürde en çok kullanılan algoritmalardan olan biyoloji tabanlı Genetik Algoritma ve sürü tabanlı Parçacık Sürü Optimizasyonu üzerine gitmeye karar verdik. Şimdi ise Genetik Algoritma ve Parçacık Sürü Optimizasyonunu daha yakından inceleyelim.

## Genetik Algoritma (GA)

Genetik algoritmalar, doğada gözlemlenen evrimsel mekanizmalara benzer mekanizmalar kullanarak çalışan eniyileştirme yöntemidir. Genetik algoritmalar problemlere tek bir çözüm üretmek yerine farklı çözümlerden oluşan bir çözüm kümesi üretir. Genetik algoritma, çözüm uzayı sonsuz olan problemlere en ideal çözüm kümelerinden birini bulmak amacıyla, 1975 yılında ‘John Holland’ tarafından ortaya atılmıştır. Bu algoritma evrimsel süreci baz almaktadır.

Bu algoritma doğayı taklit eder, örneğin bir popülasyon oluşturalım popülasyonumuzun içindeki bireyler bizim problemimizin birer çözümleri olsun, bu popülasyondaki bireylerin hepsi bir sonraki nesle geçemeyecektir sadece aralarından en güçlü, en iyi olanlar yani ortama en iyi uyum sağlayabilenler hayatta kalabilecekler. Bu popülasyondaki en iyi bireyler problemin yakınsamasını istediğimiz cevaba en yakın çözümü üretebilenlerdir. Geriye kalan bireyler sonraki nesle geçemeyeceklerdir. Yeni jenerasyonumuz da ise en iyilerden yeni çocuklar meydana gelecektir. Bu çocuklar en iyilerin genlerini taşıyacaktır. Tıpkı doğada gözlemlediğimiz gibi her yeni gelen jenerasyon bir öncekinden daha iyi bir şekilde ortama uyum sağlayacaktır, tabi ki bura da mutasyonun payını göz ardı etmemek lazım her zaman istenildiği gibi yeni jenerasyon bir öncekinden daha iyi bir uygunluk değeri sağlamayabilir ama döngünün geneline bakacak olursak ilk jenerasyondan son jenerasyona her zaman uygunluk değerinde iyileşme gözlemlenecektir.

Bu evrim sürecini şimdi matematiksel olarak ifade edecek olursak; ilk olarak problemimizi matematiksel modelini çıkarmamız gerekli. Problemimizi matematiksel olarak ifade ettikten sonra ilk popülasyonumuzu oluştururuz, popülasyon sayısı değişkendir istediğimiz sayıda belirleyebiliriz, popülasyon da ki birey sayısının az olması döngü hızını arttırırken, iterasyon sayısını da arttırmaktadır, tersi durumda ise yani popülasyon da ki birey sayısının artması durumunda da döngü hızı yavaşlarken, iterasyon sayısı da azalmaktadır. Yani bura da popülasyon sayısının seçimi önem arz etmektedir ne çok yüksek olması istenen bir durumdur ne de çok düşük olması, genel olarak algoritmalarda popülasyon sayısı ‘100’ birey olarak tanımlanmaktadır. Bu birey sayısı bizim algoritmamıza ve problemimize göre arttırılabilir de azaltılabilir de bu birazda bizim kodu çalıştıracağımız bilgisayarın gücüne bağlıdır. İlk popülasyonumuzu istediğimiz birey sayısında oluşturduktan sonra bu popülasyonumuzun her bir bireyin uygunluk değerini hesaplarız burada uygunluk değeri bizim problemimizden istediğimiz sonuca bağlıdır genellikle bir problem minimize veya maksimize edilmeye çalışılır. Her bir bireyin uygunluk değerini hesapladıktan sonra uygunluk değeri en yüksek olan bireyleri örneğin popülasyondaki en yüksek uygunluk değerine sahip olan bireylerin %10 u bir sonraki jenerasyona direk olarak aktarırız. Yeni jenerasyonumuzda ise başta belirttiğimiz popülasyondaki birey sayısına ulaşana kadar yeni çocuklar üretilecektir bu çocuklar bir önceki jenerasyonun en iyi %10 undan seçilen rasgele ebeveynlerin genlerini taşıyacaktır. Yani bu etapta ilk olarak bir ‘Seleksiyon’ yani seçilim fonksiyonu tanımlanmalıdır, bu fonksiyon bize bir adet anne ve bir adet baba döndürmelidir sonrasında bu seçilen anne ve babanın genleri bir ‘Çaprazlama’ fonksiyonu aracılığıyla çaprazlanmalı ve son olarak belirli bir mutasyon oranıyla oluşan kromozom mutasyona uğramalı veya uğramamalıdır. Sonuç olarak ortaya yeni bir birey çıktı bu birey bir önceki neslin en iyilerinin genlerini taşıyor. Bu kısımda ise özelikle dikkat etmemiz gereken şey ise mutasyon fonksiyonumuz ve mutasyon oranımızdır bunun nedeni ise tanımlayacağımız mutasyon oranına ve mutasyon fonksiyonuna göre bir sonraki jenerasyona yeni bireyler oluştururken optimal çözümden çok fazla sapabiliriz o yüzden mutasyon oranı genelde %3 civarı seçilmektedir. Tanımladığımız birey sayısına ulaşana kadar yeni bireyler oluşturduktan sonra artık yeni popülasyonumuz tanımlanmış oldu. Buradan sonrası döngü şeklinde tanımladığımız iterasyon sayısına kadar bu işlemler sürekli tekrar edecektir.

## Parçacık Sürü Optimizasyonu (PSO)

PSO, doğada sürü halinde hareket eden hayvanların yiyecek bulmak gibi temel ihtiyaçlarını giderirken sergiledikleri hareketlerin, sürüdeki diğer bireyleri etkilediğinin ve sürünün amacına daha kolay ulaştığının gözlemlenmesinden esinlenilerek Dr. Kennedy ve Dr. Eberhart tarafından 1995 yılında geliştirilmiş bir optimizasyon algoritmasıdır. PSO’da çözümü bulmak adına arama yapan her bir bireye parçacık adı verilmektedir. Bu parçacıkların tümüne ise sürü adı verilmektedir. Bir parçacığın çözüme ne kadar yakın olduğunu anlamak için tıpkı genetik algoritmada olduğu gibi bir uygunluk fonksiyonu kullanılmaktadır. PSO’ da uygunluk, parçacığın çözüme olan uzaklığı ile ölçülmektedir. Bu formül ile birlikte parçacığın yeni hızının ve yeni konumunun hesabı yapılmaktadır.

# ANTEN DİZİLERİNİN OPTİMİZASYONU

Bir anten dizisinin optimizasyonunu yaparken antenlerin birbirine göre olan konumları, faz açıları ve genlik değerleri gibi parametrelerin özenle seçilmesi gerekmektedir. Bu parametreleri bir bilgisayardan yardım almadan ayarlamak çok uzun süreler alabilmektedir. Bu yüzden günümüzde bu konuda sezgisel algoritmalardan yardım alınmaktadır. Sezgisel algoritmalar sayesinde istediğimiz sonuçlara en yakın verileri kolaylıkla çok kısa süreler içerisinde elde edebilmekteyiz.

Bu çalışmamızda meta-sezgisel algoritmaların kodlanmasın da ‘Python’ dilini kullandık. Bunun en büyük nedeni ise ‘Python’ dilinin günümüzde veri biliminde geliştiricilere çok büyük kolaylıklar sunmasıdır. Arayüz tasarımında ve geliştirilmesinde ise ‘C#’ dilini tercih ettik. Seçilmiş olan bu kodlama dilleri kişilerin kendi tercihlerine kalmaktadır. Bizim burada bu kodlama dillerini tercih etmemizin nedeni bize sunmuş oldukları kullanım kolaylıklarıdır.

Diagram, histogram

Description automatically generated

Resim .. Farklı Anten sayılarında dizi faktörü

## Genetik Algoritma ile Anten Dizilerinin Optimizasyonu

### Algoritma

Bir kodu yazmadan önce en önemli nokta algoritmanın oluşturulmasıdır. Bölüm 5’te Genetik Algoritma başlığı altında bir genetik algoritmanın nasıl çalıştığına dair kapsamlı bir bilgi vermiştik. Şimdi ise bu bilgiler ışığında algoritmamızı oluşturduk. Algoritma oluşturmanın en kolay yöntemi akış şemasını takip etmektir. Akış şeması bize hangi işlemleri hangi sırayla yapmamız gerektiğini verecektir.

A picture containing diagram

Description automatically generated

Şekil .. Genetik algoritmanın akış diyagramı

Algoritmamızın adımlarını sıradan inceleyecek olursak ilk olarak ‘Genetic’ adında bir sınıf tanımladım. Şimdi ise sırasıyla ‘Genetic’ sınıfımızın içindeki fonksiyonlara ve işleyişlerine tek tek göz atalım.

### Değişkenlerin tanımlanması

Diğer tüm algoritmalarda olduğu gibi ilk başta değişkenlerimizi tanımlamak önemlidir, özellikle programın temelini oluşturan ana değişkenler, diğer değişkenler ilerleyen dönemde programın ilerleyişine göre eklenip çıkartılabilir. Yapıcı fonksiyonumuzda burada devreye giriyor. Sınıfımız ilk çağırıldığında bu fonksiyon otomatik olarak dönmektedir bundan dolayı bu fonksiyonun içine diğer işlemlerden önce yapılmasını istediğimiz tüm işlemleri yazmalıyız. Ben bu kısımda tüm kullanacağım değişkenlerin atamasını yaptım. Bu değişkenlerden dışarıdan girdi olarak alınmaktadır. Girdi olarak alınan bu değişkenler tasarlamış olduğumuz arayüz üzerinden girdi olarak verilebilmektedir.

### Bireyler ve bireylerin kimlik bilgileri

Bu kısımda popülasyonumuza ait her bir bireyin ayrı ayrı, kromozom bilgisini, uygunluk değeri bilgisini, uygunluk değerinin toplam uygunluk değerine oranının bilgisini tuttuğumuz bir sınıf tanımladım. Yani istediğim zaman istediğim bireyin bilgilerine bu sınıf sayesinde erişebilmekteyim böylece bireyler üzerinde işlemlerimi çok daha kolay bir şekilde gerçekleştirmekteyim.

### Kod çözücü

Bu fonksiyon bizim ‘1’ ve ‘0’ lardan oluşan değişkenimizin onluk tabandaki karşılığını bulmaktadır. Fonksiyonumuz parametre olarak değişkenimizin değer aralığını, değişkenimizin bit uzunluğunu ve değişkenimizin kendisini almaktadır. Fonksiyonumuzu sırasıyla inceleyecek olursak, bu kısımda fonksiyonumuz değişken sayımız kadar dönüp her bir değişkenimiz için aynı işlemi tekrarlayacaktır. Tüm bitlerin bir olması demek değişkenimizin tanım aralığındaki en büyük değer demektir, tüm bitlerimizin sıfır olması ise değişkenimizin tanım aralığındaki en küçük sayı demektir diğer durumlar ise bu koşula göre işlenmektedir. Bu kısımda bit sayısı ne kadar fazla ise hassasiyette o kadar fazla olacaktır. Bit sayısı yine arayüz üzerinden kullanıcı tarafından seçilebilmektedir.

### Dizi faktörü

Bu fonksiyonumuzun içinde lineer anten dizilerine ait dizi faktörü denklemimizi tanımlarız, bu fonksiyon parametre olarak antenlerin faz farklarının değer aralığını, antenlerinin genliklerinin değer aralığını ve antenlerin konumlarının değer aralığını alır bu parametrelerin fonksiyona gönderilmeden önce kod çözücü fonksiyonuna tabi tutulması önemlidir.

### Gen üretici

Bu fonksiyonumuzun görevi; çağırıldığında bize ‘0’ veya ‘1’ değerini rasgele olarak döndürmesidir.

### Kromozom üretici

Bu fonksiyonumuzun görevi; çağırıldığında bize bit sayımız kadar Gen Üretici fonksiyonunu çağırarak bir adet kromozom döndürmesidir.

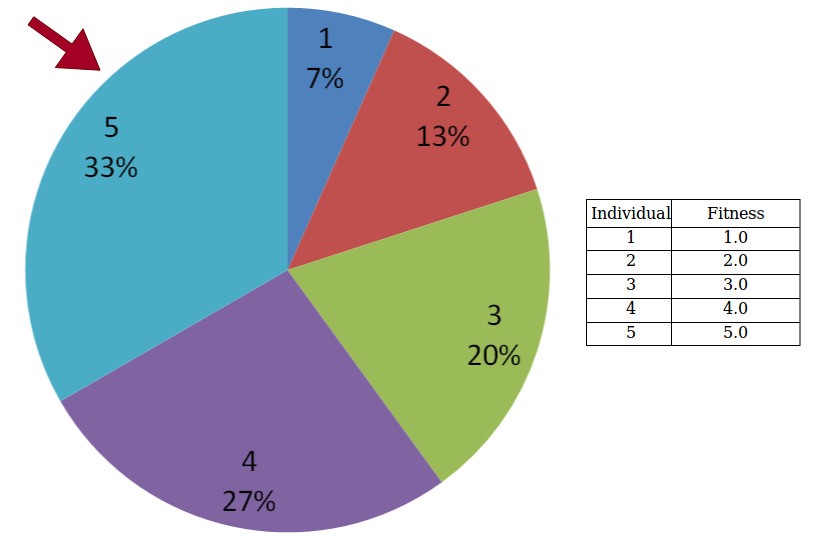
### Uygunluk değerinin hesaplanması

Bu fonksiyonumuz algoritmamızın bel kemiği desek yanlış olmaz çünkü istediğimiz sonucu alabilmemiz için uygunluk değerinin doğru hesaplanması önem arz etmektedir.

### Seleksiyon işlemi

Bu fonksiyonumuzda çaprazlama işleminin gerçekleştirilmesi ve yeni bireyin oluşturulması için ebeveyn seçme işlemi yapılmaktadır. Bu işlemi ben ‘rulet tekerleği’ yöntemi ile gerçekleştirdim. Bu yöntemden kısaca bahsedecek olursak; bir çarkıfelek mantığı ile çalışır, bireyler uygunluk değerlerinin büyüklüğüne göre çarkıfeleğin üzerinde yer kaplamaktadır yani her bireyin seçilme ihtimali vardır fakat bireyin uygunluk değeri ne kadar yüksek ise seçilme ihtimalide bir o kadar yüksek olacaktır.

Table

Description automatically generated

Şekil .. Rulet tekerleği yöntemi

İlk olarak uygunluk değerlerimizin toplam değerini tutacak bir değişken tanımladım sonrasında bir de toplam oranı tutacak bir değer tanımladım yani ‘%100’. Şimdi ise içerisinde bir önceki jenerasyonun en iyilerini ve en iyilerden üretilen yeni bireyleri tutan listenin içerisinde dönerek her bir bireyin uygunluk değeri tek tek toplam uygunluk değeri değişkenimize eklenir. Sonrasında ise yine aynı listenin içinde dönerek her bir bireyin kendi uygunluk değerinin toplam uygunluk değerine oranının yüzdesi alınır ve bireye atanır. Şimdi ise rasgele olarak sıfır ile yüz arasında bir zar atılır sonrasında yeni jenerasyonumuzdan oluşan listenin içerisinde tek tek dönülmeye başlanır, n’inci bireyimizin uygunluk değeri yüzdesi o anki toplam uygunluk değeri yüzdesinden çıkartılır ve attığımız zardan küçük olup olmaması kontrol edilir eğer ki koşul sağlanırsa o birey birinci ebeveyn olarak atanır aynı işlem tekrar yapılır ve ikinci ebeveyn seçilir. Bu işlemde her bir bireyin seçilme ihtimali vardır ama bireyin uygunluk değeri ne kadar yüksek ise seçilme ihtimalide bir o kadar fazla olacaktır. Bu yöntem ile birlikte her yeni nesilin bir önceki nesilden daha iyi olması hedeflenmektedir.

### Çaprazlama işlemi

Bu fonksiyonumuz çaprazlama işleminin gerçekleştiği fonksiyondur, parametre olarak çaprazlama işleminin gerçekleşme oranını almaktadır. Bu fonksiyonun işleyişini kodun üzerinde inceleyelim. İlk olarak popülasyonumuzdaki uygunluk değeri en yüksek olan %10’luk kesimi sonraki jenerasyonun tutulduğu listemizin içine atıyoruz. Sonrasında ise yeni jenerasyonumuzda kalan eksik kısmı en iyi %10’dan oluşan bireylerden üreteceğimiz yeni bireylerle dolduracağız ve birey sayımız tanımladığımız popülasyon uzunluğuna ulaşınca döngümüzü keseceğiz. İlk olarak ‘seleksiyon fonksiyonunu çağırıyoruz ve bu fonksiyon bize iki adet ebeveyn döndürecektir. Sonrasında döndürülen ebeveynleri çaprazlama işlemine sokmadan önce bir koşula sokuyoruz bu koşul şu şekildedir sıfır ile bir arasında bir zar atılır ve atılan zar tanımlamış olduğumuz çaprazlama işleminin gerçekleşme oranından küçük ise çaprazlama işlemi gerçekleşecektir eğer koşul sağlanmaz ise direkt seçilen bireyler tekrardan çaprazlanmadan yeni jenerasyona yeni bireyler olarak atanacaktır. Şimdi ise koşulun sağlandığı duruma bakacak olursak, bir ile kromozom uzunluğunun iki eksiği kadar olan sayı arasında bir zar atılır, zarın sonucuna x diyelim, birinci yeni bireyimiz birinci ebeveynin x e kadar olan kısmını alıp ikince ebeveynden de x den sonrasını alırken, ikinci yeni bireyimizde birinci ebeveynin x den sonrası alınırken ikinci ebeveynin x den öncesi alınmaktadır. Sonrasında bu yeni bireyler tek tek mutasyon fonksiyonuna gönderilerek mutasyon işlemine tabi tutulurlar ve yeni jenerasyon listemize eklenirler. Bu işlem yeni jenerasyonumuzdaki birey sayısı tanımlamış olduğumuz popülasyondaki olması gereken birey sayısına eşit olana kadar tekrarlanır ve son olarak döngü tamamlandıktan sonra yeni jenerasyona atadığımız yeni bireyler popülasyon listemize aktarılır.

Chart

Description automatically generated with medium confidence

Şekil .. Noktasal çaprazlama yöntemi

### Mutasyon işlemi

Bu fonksiyonda mutasyon işlemi gerçekleştirilmektedir. Fonksiyonumuz iki adet parametre almaktadır bunlardan ilki bireyin kromozom bilgisi ikincisi ise mutasyon işleminin gerçekleşme oranıdır. Fonksiyonumuzun çalışması koşula bağlıdır koşulumuz ise atmış olduğumuz zarın tanımlamış olduğumuz mutasyonun gerçekleşme oranından küçük gelmesidir. Fonksiyonumuzu inceleyelim. Fonksiyonumuz şu şekilde işlemektedir, bireyimizin içinde döneriz sonrasında her adımda bir zar atarız eğer attığımız zar mutasyon gerçekleşme değerinden daha küçük gelirse kromozomumuzun o adımdaki değeri bir ise sıfır, sıfır ise bir yapılır.

### Genetik algoritma

Bu kısımda ise tanımlamış olduğumuz fonksiyonları belirli bir hiyerarşi içerisinde tek tek çağıracağız yani bu bizim sınıfımızın ana fonksiyonudur.

## PSO ile Anten Dizilerinin Optimizasyonu

### Algoritma

Bir kodu yazmadan önce en önemli nokta algoritmanın oluşturulmasıdır. Bölüm 7’de Parçacık Sürü Optimizasyonu başlığı altında bir PSO algoritmasının nasıl çalıştığına dair kapsamlı bir bilgi vermiştik. Şimdi ise bu bilgiler ışığında algoritmamızı oluşturduk. Algoritma oluşturmanın en kolay yöntemi akış şemasını takip etmektir. Akış şeması bize hangi işlemleri hangi sırayla yapmamız gerektiğini verecektir.

A picture containing diagram

Description automatically generated

Şekil .. Parçacık sürü optimizasyonunun akış diyagramı

Algoritmamızın adımlarını sıradan inceleyecek olursak ilk olarak ‘PSO’ adında bir sınıf tanımladım. Şimdi ise sırasıyla ‘PSO’ sınıfımızın içindeki fonksiyonlara ve işleyişlerine tek tek göz atalım.

### Değişkenlerin tanımlanması

Genetik algoritma ve diğer tüm algoritmalarda olduğu gibi ilk başta değişkenlerimizi tanımlamak önemlidir. Sınıfımız ilk çağırıldığında bu fonksiyon otomatik olarak dönmektedir bundan dolayı bu fonksiyonun içine diğer işlemlerden önce yapılmasını istediğimiz tüm işlemleri yazmalıyız. Ben bu kısımda arayüzden girdi olarak aldığımız temel değişkenleri tanımladım.

### Parçacıklar ve parçacıkların kimlik bilgileri

Bu kısımda sürümüze ait her bir parçacığın ayrı ayrı kimlik bilgilerini tutmaktayız. Bunlar; parçacığın o anki konumu, parçacığın tüm zamanlarda çözüme en çok yaklaştığı konumu, parçacığın şu anki uygunluk değeri, parçacığın tüm zamanlardaki en yüksek uygunluk değeri ve parçacığın hızıdır.

### Dizi faktörü

Bu fonksiyonumuzun içinde lineer anten dizilerine ait denklemimizi tanımlarız, bu fonksiyon parametre olarak antenleri faz farklarının değer aralığını, antenlerinin genliklerinin değer aralığını ve antenlerin konumlarının değer aralığını alır.

### Parçacık üretici

Bu fonksiyonumuz, sürüye rasgele olarak yeni parçacıklar atamaktadır.

### Parçacıkların hız ve konum bilgilerinin güncellenmesi

Bu fonksiyonumuz, her jenerasyonda parçacıkların yeni konumlarını belirlememizi sağlamaktadır.

### Uygunluk değerinin hesaplanması

Bu fonksiyonumuz, sürüdeki parçacıkların optimal çözüme olan uzaklıklarını yani uygunluk değerlerini hesaplayıp, parçacıkları puanlandırmaktadır.

### Parçacık sürü optimizasyonu

Bu kısımda tanımlamış olduğumuz fonksiyonları belirli bir hiyerarşi içerisinde tek tek çağıracağız. Fonksiyonumuz tanımlamış olduğumuz iterasyon sayısı kadar dönecektir. Fonksiyonumuz son olarak ise bizlere gerekli olan grafik çizimlerini yapacaktır.

# UYGULAMA HAKKINDA

Bu kısımda kullanmış olduğunuz uygulamaya ait bilgiler sizlere sunulmuştur. Uygulamayı daha yakından tanımak için bu yazıdan yararlanabilirsiniz.

## Arayüzün Genel Yapısı

Arayüzün genel yapısında kullanıcıların önüne farklı menüler arasında kolaylıkla geçişler yapabileceği bir menü gelmektedir. Aynı zamanda üst kısımda yer alan araç çubuğu ile birlikte kullanıcılar yapmak istedikleri işlemleri çok daha hızlı ve pratik bir şekilde gerçekleştirebilmektedir.



Resim .. Arayüzün ana sayfası

Resim 6.1’de gördüğümüz programın ilk açıldığında karşımıza çıkan ekran ‘Ana Menü’ sayfasıdır. Arayüzün üst kısmında araç çubuğu yer alırken sol kısmında ise sayfalar arası geçiş yapabileceğimiz menü yer almaktadır. Menüde farklı sayfaların yer aldığını görmekteyiz. Bu sayfalardan ilki ‘Ana Menü’ sayfasıydı, diğer sayfaları da sırasıyla ayrıntılı bir şekilde inceleyelim.

## Antenler Hakkında Kaynak

Bu sayfamızda kullanıcılara antenler hakkında kapsamlı bir kaynak sunmaktayız. Kullanıcılar bu sayfa aracılığıyla optimizasyon işlemlerini yapmanın yanı sıra antenler hakkında da bilgi edinebilecekler. Bu sayfanın arka planında PDF görüntüleyiciye benzer bir yapı çalışmaktadır ve bu yapı aracılığıyla yapmış olduğum araştırmalar sonucu elde ettiğim bilgileri kullanıcılara aktarabilmekteyim.



Resim .. Anten sayfası

## Yapay Zekâ Hakkında Kaynak

Bu sayfamızda ‘Anten’ sayfamız ile benzer yapıdadır. Bu sayfamızda ‘Anten’ sayfamızdan farklı olarak antenler yerine yapay zekâ hakkında kullanıcılara kapsamlı bir kaynak sunmaktadır. Bu bilgilere ek olarak arka planda çalışan optimizasyon algoritmalarına da bu kaynakta yer verilmiştir. Yine bu sayfamızın arka planında da PDF görüntüleyiciye benzer bir yapı çalışmaktadır ve bu yapı aracılığıyla yapmış olduğum araştırmalar sonucu elde ettiğim bilgileri kullanıcılara aktarabilmekteyim.

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

Resim .. Yapay zekâ sayfası

## Optimizasyon İşlemlerinin Gerçekleştirilmesi

Bu sayfa projemizin kendisini oluşturmaktadır. Bu sayfa ile birlikte kullanıcılar farklı yapay zekâ algoritmaları ile farklı dizi anten topolojilerini kolaylıkla optimize edebilmektedir. Şu anlık sadece lineer anten dizileri optimize edilebilmektedir. Bu sayfayı açtığımızda karşımıza ilk olarak farklı sezgisel algoritmaları seçebildiğimiz bir sayfa karşılamaktadır bizleri. Bu kısımda ilk etapta Genetik Algoritma ve Parçacık Sürü Optimizasyonu algoritmaları kullanıcılara seçenek olarak sunulmuştur. İlerleyen süreçte algoritma ve dizi anten topoloji sayısı arttırılması hedeflenmektedir.

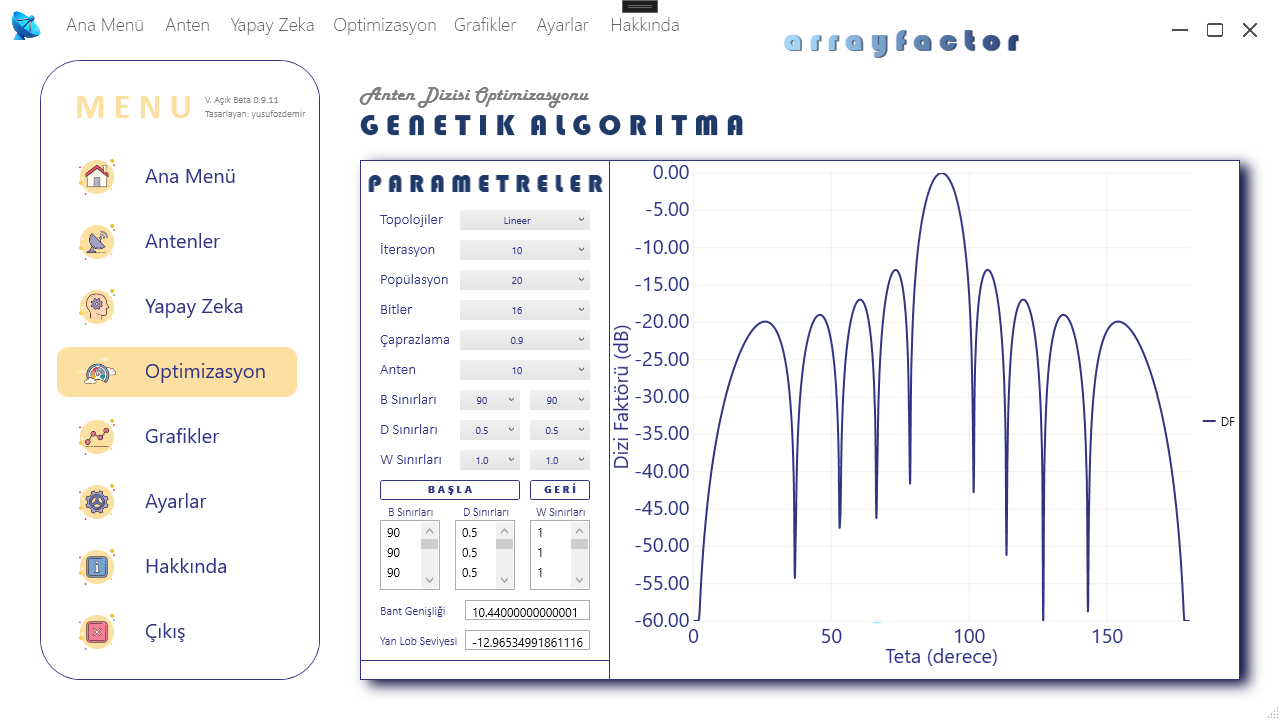
diyagram içeren bir resim

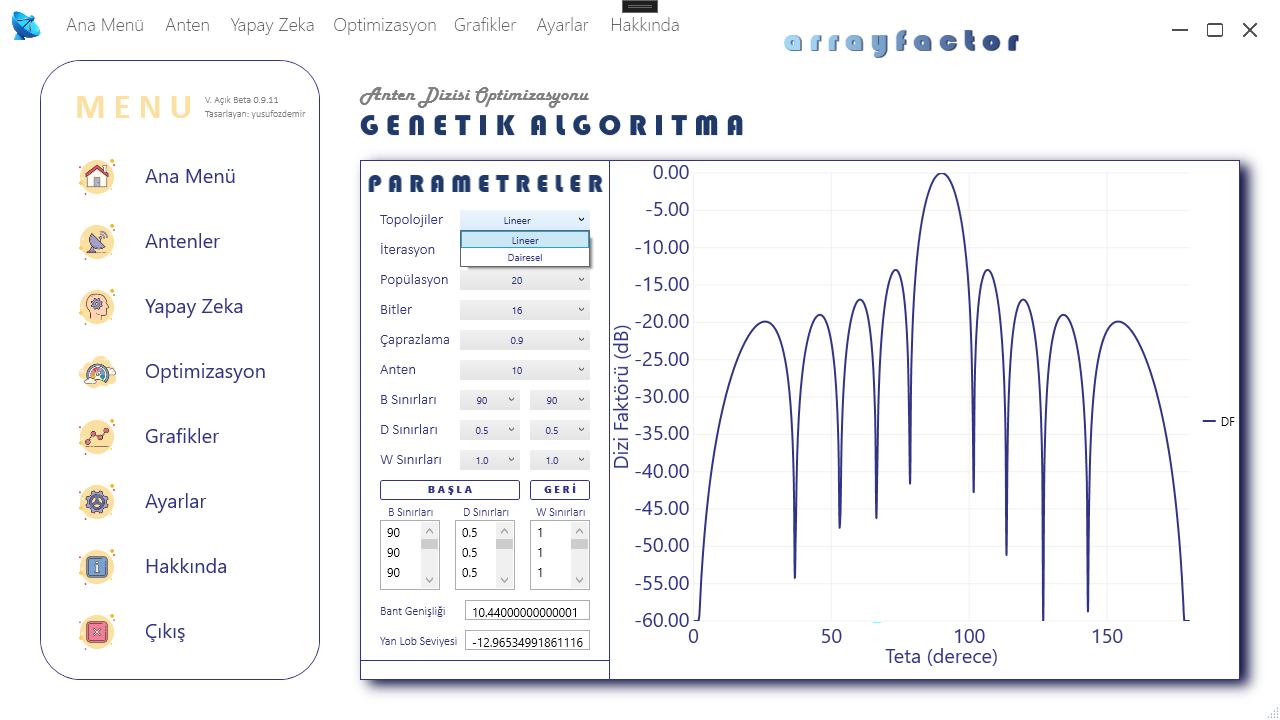
Açıklama otomatik olarak oluşturuldu

Resim .. Optimizasyon sayfası

### Genetik algoritma ile optimizasyon işlemleri

Kullanıcı eğer ‘GA’ butonuna tıklayacak olursa karşısına ‘Genetik Algoritma’ paneli çıkacaktır. Bu panel iki kısımdan oluşmaktadır. Panelin sol üst kısmından gerekli parametre ayarlamaları yapılabilirken sol alt kısımda ise programın çalıştırıldıktan sonraki bize verdiği çıktıların görüntüleneceği listeler yer almaktadır. Panelin sağ kısmında ise tasarlanan anten dizisinin ışıma örüntüsü kullanıcılara sunulmaktadır.

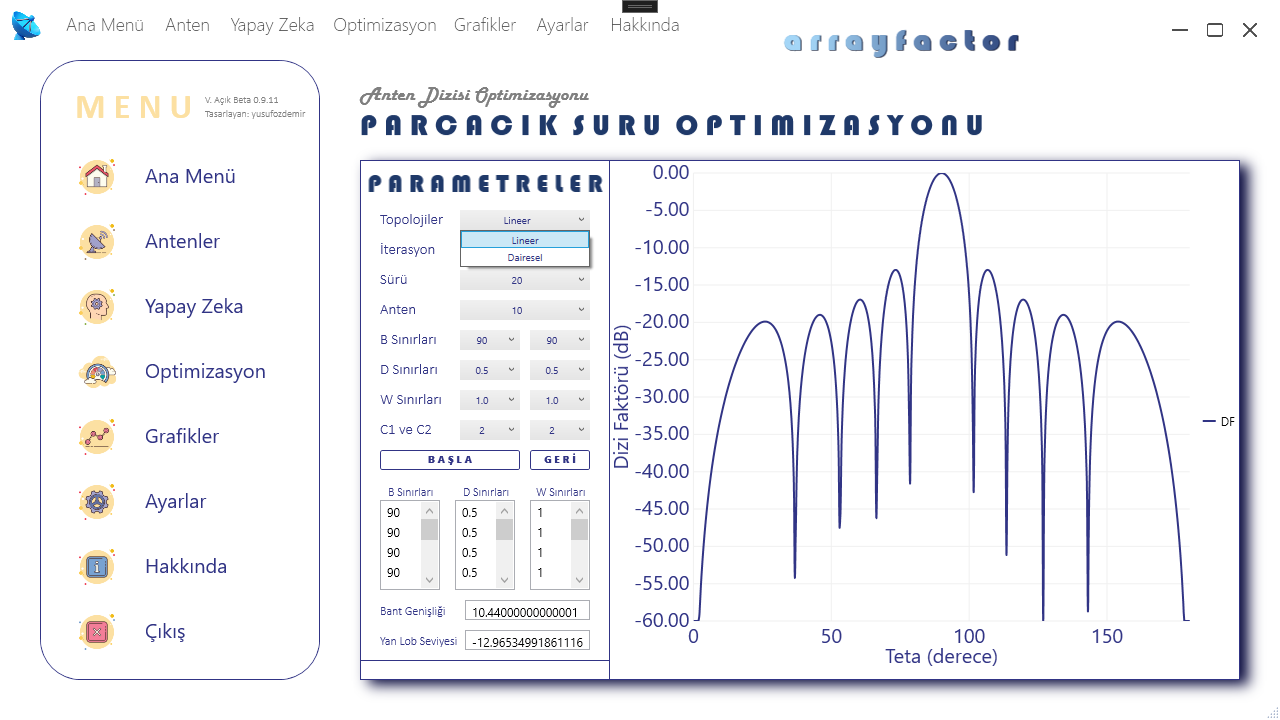
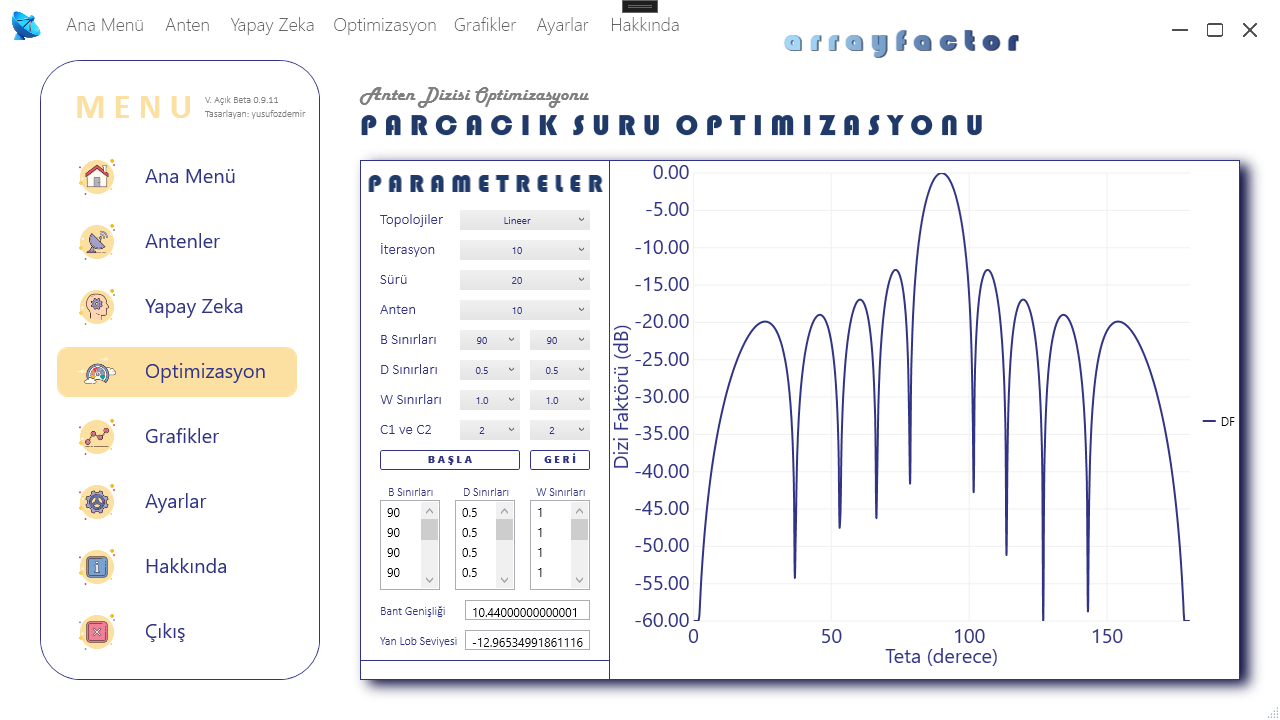




Resim .. Genetik algoritma paneli

### Parçacık sürü optimizasyonu ile optimizasyon işlemleri

Kullanıcı eğer ‘PSO’ butonuna tıklayacak olursa karşısına ‘Parçacık Sürü Optimizasyonu’ paneli çıkacaktır. Bu panelin yapısı Genetik Algoritma paneli ile birebir aynıdır. Aralarındaki tek fark algoritmaların içerdiği farklı parametre ayarlamalarıdır.



Resim .. Parçacık sürü optimizasyonu paneli

## Grafik Karşılaştırma İşlemleri

Bu panel aracılığı ile kullanıcılar optimize ettikleri farklı anten dizilerini veri tabanından çekerek ışıma örüntülerinin karşılaştırmasını kolaylıkla yapabilmektedirler.

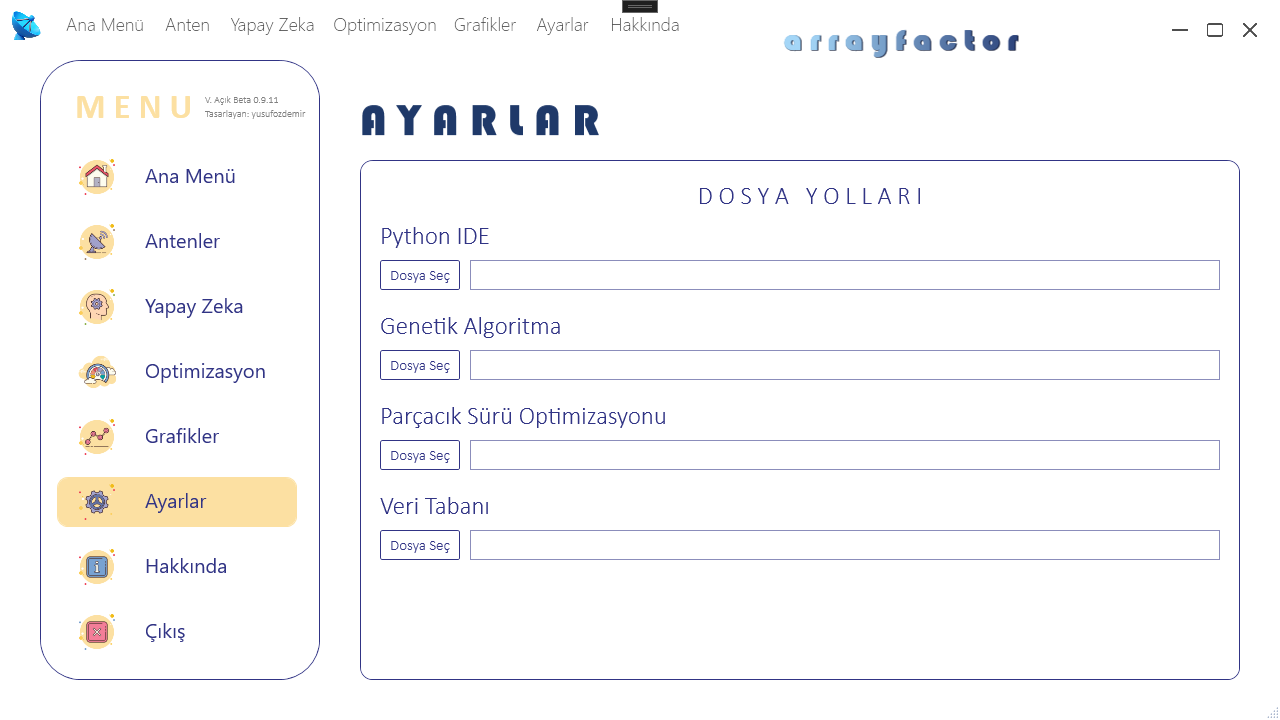
çizelge içeren bir resim

Açıklama otomatik olarak oluşturuldu

Resim .. Grafikler paneli

## Programın Çalışabilmesi İçin Gerekli Ayarlamaların Yapılması

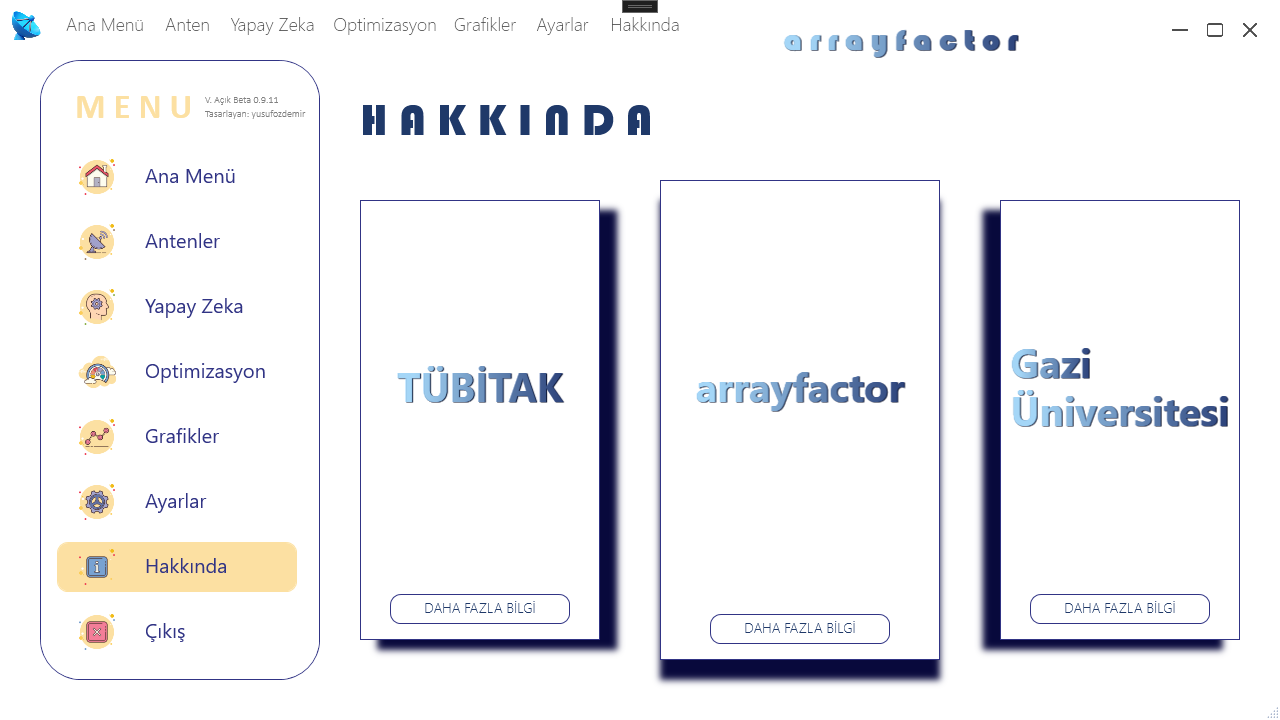
Bu sayfada kullanıcılara gerekli olan dosya yollarını seçmeleri için butonlar atanmıştır. Arka planda çalışan yapay zekâ algoritmaları Python ile kodlanmıştır ve arayüz bu algoritmaları çalıştırabilmek için Python derleyicisine ihtiyacı vardır. Kullanıcı kendi bilgisayarında yer alan Python derleyicisini ‘Python Derleyici Dosya Seç’ butonuna tıklayarak seçebilmektedir. Buna ek olarak kullanıcının kendi algoritmalarını da yükleyebilmesi için algoritmaları seçebileceği butonlarda eklenmiştir. Bu sayede kullanıcı kendi geliştirdiği algoritmaları bu sayfadan dosya yolunu seçerek arayüz üzerinden optimizasyon testlerini yapabilecektir. Son olarak ise kullanıcı veri tabanının kayıt altına alınmasını istediği dosya yolunu yine bu sayfadan seçebilmektedir.



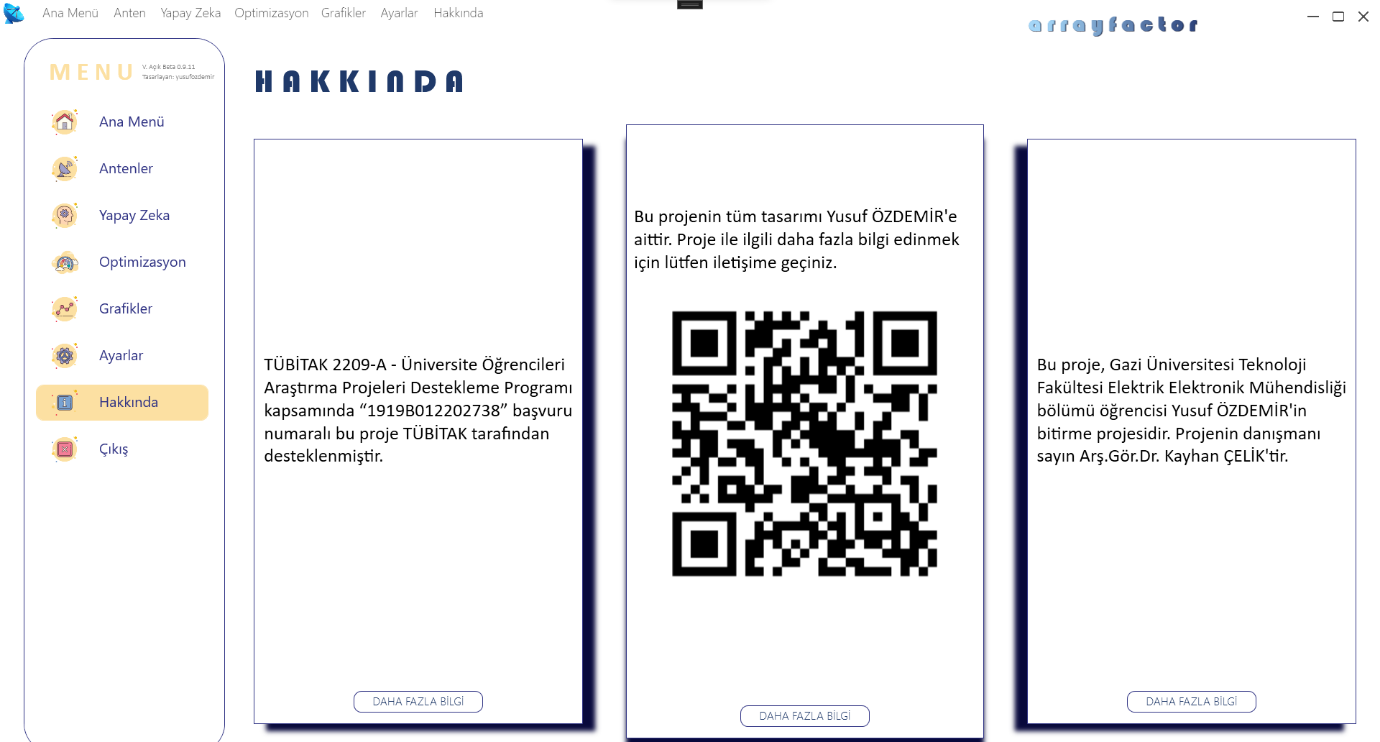
Resim .. Ayarlar sayfası

## Proje Hakkında Bilgiler

Bu sayfada proje hakkında bilgiler kullanıcılara üç başlık altında sunulmuştur. Bunlar; projemizi destekleyen ‘TÜBİTAK’, projenin yapıldığı kurum ‘Gazi Üniversitesi’ ve projenin başlığı olan ‘arrayfactor’. Bu başlıklar altında kullanıcılara kısa bilgiler vermek amacıyla paneller oluşturulmuştur.



Resim .. Hakkında sayfası görünüm 1



Resim .. Hakkında sayfası görünüm 2

## Veri Tabanı

Bu kısım arayüzde yer almamaktadır. Arayüzümüzde yaptığımız optimizasyonların sonuçları ayarlardan seçmiş olduğumuz dosya yoluna ‘Excel’ dosyası olarak kayıt altına alınmaktadır. Kullanıcılar bu sayede istediği zaman istediği optimizasyon sonuçlarına kolaylıkla erişebilmektedir. Aşağıda bu ‘Excel’ dosyasının yapısı yer almaktadır.

masa içeren bir resim

Açıklama otomatik olarak oluşturuldu

Resim .. Veri tabanında veriler

Arayüz tarafından oluşturulan ‘Excel’ dosyasının A, B ve C sütunlarında arayüz üzerinden seçmiş olduğumuz parametrelerin değerleri yer almaktadır. D, E, F, G, H, I ve J sütunlarında ise optimizasyon sonucu elde edilen sonuçlar yer almaktadır. Bunlara ek olarak sonuçların grafiksel çıktısı da yine ‘Excel’ dosyasında kullanıcılara sunulmaktadır.

diyagram içeren bir resim

Açıklama otomatik olarak oluşturuldu

Resim . Veri tabanında görselleştirilmiş veriler

# OPTİMİZASYON SONUÇLARI

Arayüz üzerinden yapmış olduğumuz birkaç optimizasyonun sonucunu birlikte inceleyelim. İlk olarak tekrardan optimizasyon sonucunda hedeflediğimiz dönütleri ele alacak olursak, anten dizilerinin optimizasyonunda yapılan optimizasyonun sonucunda SLL (Yan Lob Seviyesi), HPBW (Yarım Güç Bant Genişliği), yönlülük ve boş kontrol parametrelerinde iyileştirmeler hedeflenmektedir. Yan bantlar bastırılırken, yarım güç bant genişliğinde daralma meydana gelmesi ana hedefimizdir. Biz optimizasyon işlemlerimizi sezgisel algoritmalar ile gerçekleştirdik. Sezgisel algoritmalar ile çalıştırılma sürelerine bağlı olarak çok daha iyi sonuçlar elde etmek mümkündür. Elimizde mevcut olan giriş seviye bir bilgisayar ve kısa süreli bir çalışma sonucu elde etmiş olduğumuz sonuçları şimdi beraber inceleyelim.

Diagram, histogram

Description automatically generated

Resim .. Örnek optimizasyon sonucu 1

A picture containing histogram

Description automatically generated

Resim .. Örnek optimizasyon sonucu 2

Üst kısımda yer alan grafiklerde görebileceğimiz üzere hedeflerimizden biri olan yan bantlar bastırılmıştır fakat yarım güç bant genişliğinde daralma meydana gelmemiştir. Bunun nedeni üst kısımda da bahsettiğimiz gibi tamamen algoritmanın çalıştırılma süresi ile alakalıdır. Güçlü bir bilgisayarda yüksek popülasyon ve yüksek iterasyon sayısı altında algoritmamızı uzun süreler çalıştıracak olursak çok daha iyi sonuçlar elde etmemiz mümkün olacaktır.