

Genetik Algoritma ile
kullanılarak

FABRİKA MAKİNE OPTİMİZASYON PROJESİ

Seydi Ali İçlek

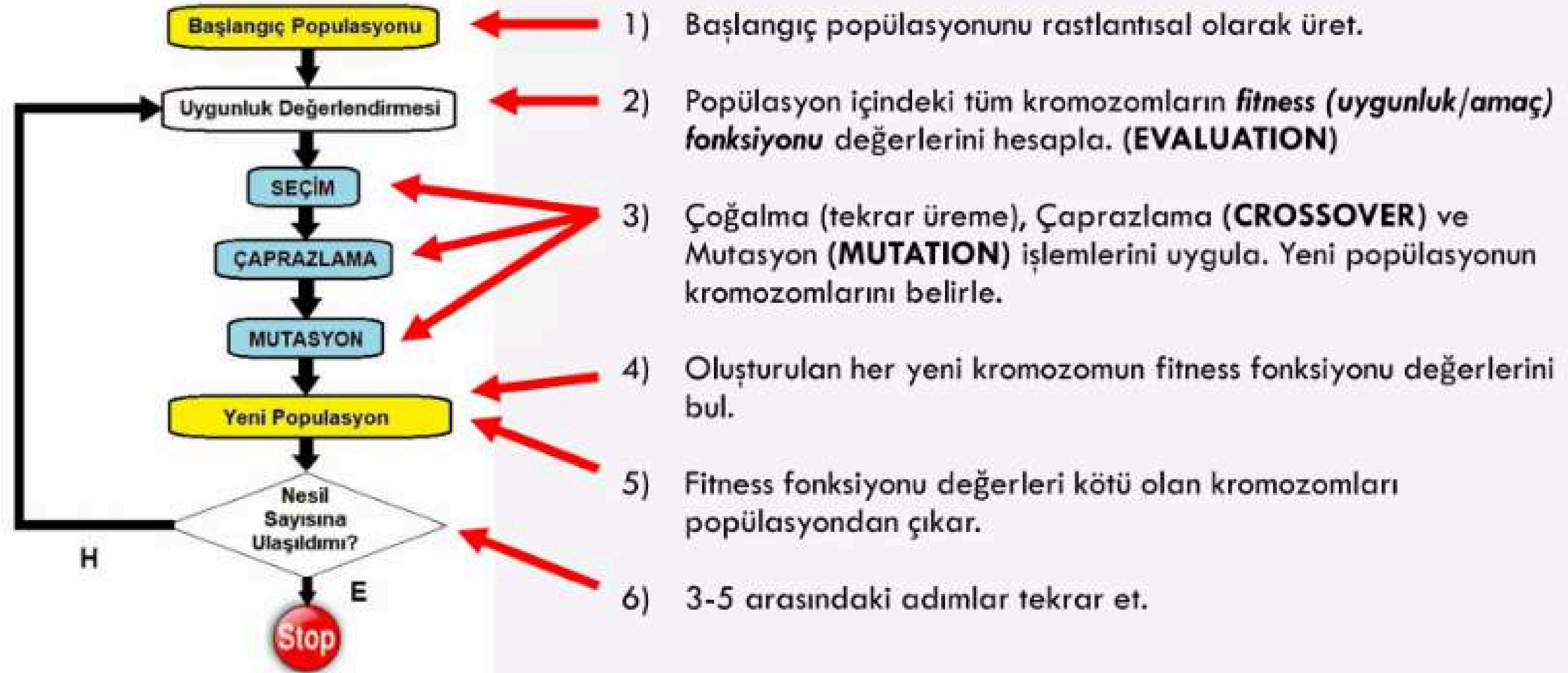
Genetik Algoritma Nedir?

Genetik Algoritma (GA), doğadaki evrimsel süreçlerden ve özellikle Charles Darwin'in doğal seçim teorisinden ilham alınarak geliştirilmiş, sezgisel arama ve optimizasyon yöntemidir.

Bu algoritmalar, biyolojik evrimde gözlemlenen doğal seçim, genetik çaprazlama ve mutasyon gibi temel mekanizmaları kullanarak, karmaşık problemler için **en iyiye yakın** çözümleri bulmayı hedefler.

Özellikle karmaşık problemlerin belirli bir matematiksel modelle ifade edilemediği veya geleneksel optimizasyon yöntemlerinin yetersiz kaldığı durumlarda ortaya çıkar.

Genetik Algoritma Aşamaları



Genetik Algoritma Aşamaları

1. Başlangıç Popülasyonu: Olası çözümlerden oluşan rastgele bir başlangıç kümesi oluşturulur.
2. Uygunluk Değerlendirmesi: Her çözümün probleme ne kadar "iyi" uyum sağladığı bir uygunluk fonksiyonu ile ölçülür.
3. Seçilim: Daha "uygun" çözümlerin sonraki nesli oluşturmak üzere seçilme olasılığı artırılır.
4. Çaprazlama & Mutasyon: Seçilen çözümler arasında genetik bilgi alışverişi (çaprazlama) ve rastgele küçük değişiklikler (mutasyon) yapılarak yeni çözümler üretilir.
5. Tekrarlama: Süreç, belirli bir durdurma kriterine ulaşılan kadar tekrarlanır.

Genetik Algoritma Aşamaları

1. Başlangıç Popülasyonu:

python

```
def birey_olustur():
    # Önce her türe 1 makine ver
    birey = [1 for _ in makine_verisi]
    kalan_makineler = toplam_makine - len(makine_verisi)

    if kalan_makineler < 0:
        raise ValueError(f"Toplam makine sayısı en az {len(makine_verisi)} olmalıdır!")

    # Kalan makineleri rastgele dağıt
    if kalan_makineler > 0:
        ekstra = [random.randint(0, kalan_makineler) for _ in makine_verisi]
        toplam_ekstra = sum(ekstra)
        if toplam_ekstra > 0:
            normalize_ekstra = [int(round(x * kalan_makineler / toplam_ekstra)) for x in ekstra]
            normalize_ekstra[-1] = kalan_makineler - sum(normalize_ekstra[:-1])
            birey = [taban + ekstra for taban, ekstra in zip(birey, normalize_ekstra)]

    return birey

# Başlangıç popülasyonunu oluşturma
arac_kutusu.register("birey", tools.initIterate, creator.Birey, birey_olustur)
arac_kutusu.register("populasyon", tools.initRepeat, list, arac_kutusu.birey)
populasyon = arac_kutusu.populasyon(n=POPULASYON_BOYUTU) # 100 bireylik popülasyon
```

Genetik Algoritma Aşamaları

2. Uygunluk Değerlendirilmesi:

Bu fonksiyon her çözümün uygunluğunu değerlendirir.
Burada amaç toplam süreyi minimize etmek.

```
def degerlendir(birey):  
    # Her türden en az 1 makine olmalı  
    if any(sayi < 1 for sayi in birey) or sum(birey) != toplam_makine:  
        return float('inf'),  
  
    toplam_sure = 0  
    for i, sayi in enumerate(birey):  
        islem_suresi = makine_verisi[i]['islem_suresi']  
        bekleme_suresi = makine_verisi[i]['bekleme_suresi']  
        toplam_sure += (islem_suresi + bekleme_suresi) * sayi  
    return toplam_sure,  
  
arac_kutusu.register("degerlendir", degerlendir)
```

Genetik Algoritma Aşamaları

3. Seçim:

Her seferinde rastgele 3 birey seçilip en iyisi alınıyor.

```
arac_kutusu.register("select", tools.selTournament, tournsize=3)  
populasyon = arac_kutusu.select(populasyon, len(populasyon))
```

Genetik Algoritma Aşamaları

3. Çaprazlama:

Her seferinde rastgele 3 birey seçilip en iyisi alınıyor.

```
arac_kutusu.register("mate", tools.cxTwoPoint) # İki noktalı çaprazlama
```


Genetik Algoritma Aşamaları

3. Mutasyon:

Her seferinde rastgele 3 birey seçilip en iyisi alınıyor.

```
def ozel_mutasyon(birey, olaslik):
    for i in range(len(birey)):
        if random.random() < olaslik:
            mevcut_deger = birey[i]
            degisim = random.randint(-2, 2)
            yeni_deger = max(1, mevcut_deger + degisim)
            birey[i] = yeni_deger

    # Toplam makine sayısını koru
    mevcut_toplam = sum(birey)
    if mevcut_toplam != toplam_makine:
        fark = toplam_makine - mevcut_toplam
        ayar_indeks = random.randint(0, len(birey) - 2)
        birey[ayar_indeks] = max(1, birey[ayar_indeks] + fark)

    return birey,

arac_kutusu.register("mutate", ozel_mutasyon, olaslik=0.2)
```

Genetik Algoritma Aşamaları

3. Tekrarlama:

Her seferinde rastgele 3 birey seçilip en iyisi alınıyor.

```
# Her nesil için istatistikleri toplama
for nesil in range(MAKS_NESIL):
    populasyon = algorithms.varAnd(populasyon, arac_kutusu, CAPRAZLAMA_OLASILIK, MUTASYON_OLASILIK)

    # Uygunluk değerlerini hesaplama
    uygunluklar = arac_kutusu.map(arac_kutusu.degerlendir, populasyon)
    for uygunluk, birey in zip(uygunluklar, populasyon):
        birey.fitness.values = uygunluk

    # Yeni nesil için seçim
    populasyon = arac_kutusu.select(populasyon, len(populasyon))

    # İstatistikleri kaydetme
    nesil_istatistik = istatistik.compile(populasyon)
    nesiller.append(nesil)
    min_uygunluk.append(nesil_istatistik['min'])
    ort_uygunluk.append(nesil_istatistik['ort'])
```


Genetik Algoritma Aşamaları

Döngü

Bu döngü 50 nesil boyunca devam ediyor. Her nesilde:

1. Popülasyona çaprazlama ve mutasyon uygulanıyor
2. Yeni bireylerin uygunluk değerleri hesaplanıyor
3. Bir sonraki nesil için bireyler seçiliyor
4. İlerlemeyi takip etmek için istatistikler kaydediliyor

Sonuç

En sonunda, en iyi çözüm bulunuyor:

```
en_iyi_birey = tools.selBest(populasyon, k=1)[0]  
en_iyi_uygunluk = en_iyi_birey.fitness.values[0]
```


Uygulama

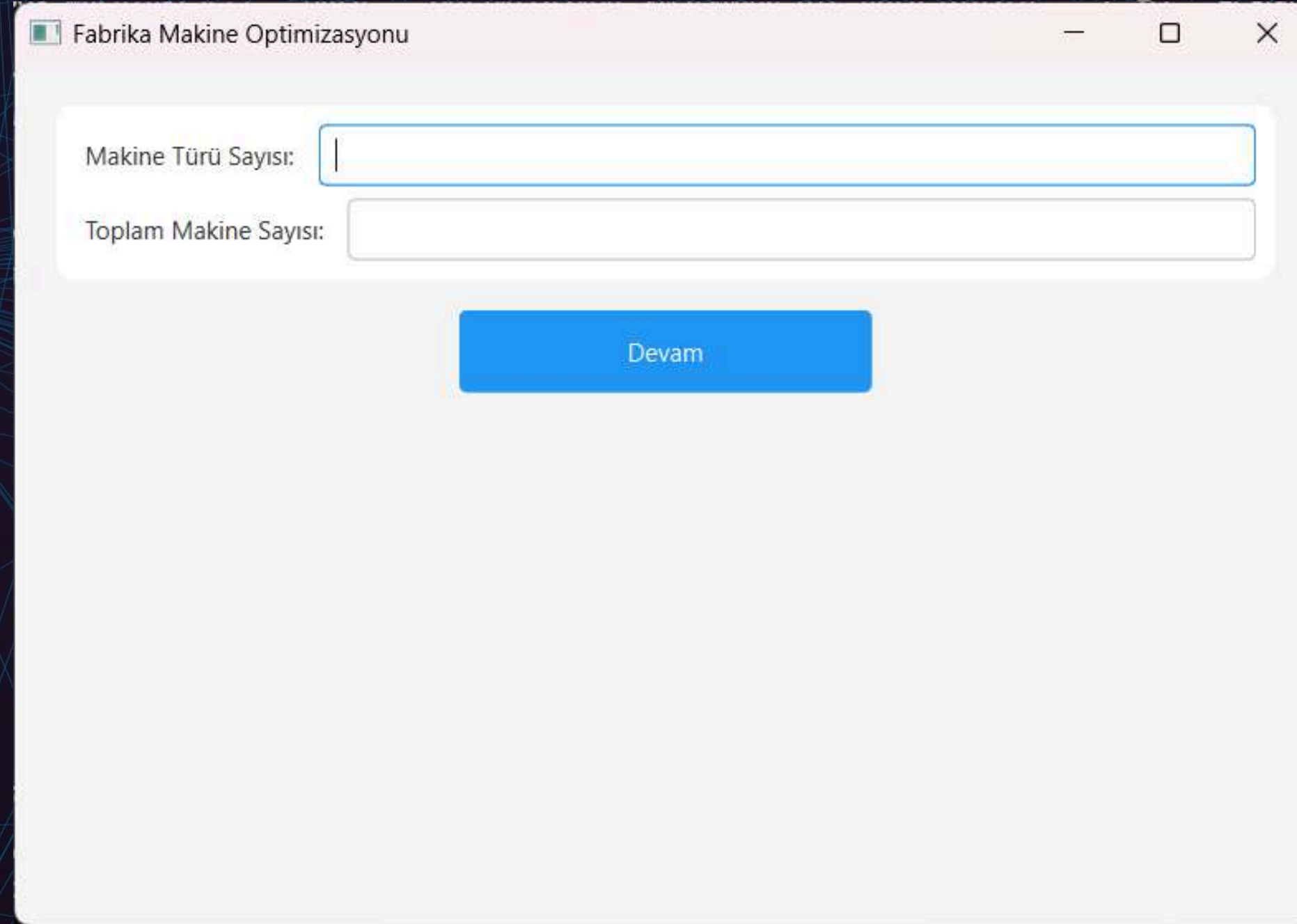
PyQt6 bir GUI (Grafiksel Kullanıcı Arayüzü) kütüphanesidir.

NumPy (Numerical Python), Python'da bilimsel hesaplamalar için kullanılan kütüphanedir.

DEAP (Distributed Evolutionary Algorithms in Python), Python'da evrimsel algoritmalar ve genetik programlama için kullanılan bir frame'dir.

Matplotlib kütüphanesinin bir modülüdür ve grafik çizmek için MATLAB benzeri bir arayüz sağlar. Çeşitli grafik ve çizimler oluşturmak için kullanılır.

Uygulama



Fabrika Makine Optimizasyonu

Makine Türü Sayısı:

Toplam Makine Sayısı:

Devam

Makine türü ve toplam sayısı yazılır

Uygulama

Fabrika Makine Optimizasyonu

Makine Türü Sayısı: 4

Toplam Makine Sayısı: 50

Devam

Makine Türü 1 Adı: Kesim

İşlem Süresi (saniye): 50

Bekleme Süresi (saniye): 10

Makine Türü 2 Adı: Kaynak

İşlem Süresi (saniye): 70

Bekleme Süresi (saniye): 15

Makine Türü 3 Adı: Montaj

İşlem Süresi (saniye): 100

Bekleme Süresi (saniye): 5

Makine Türü 4 Adı: Paketleme

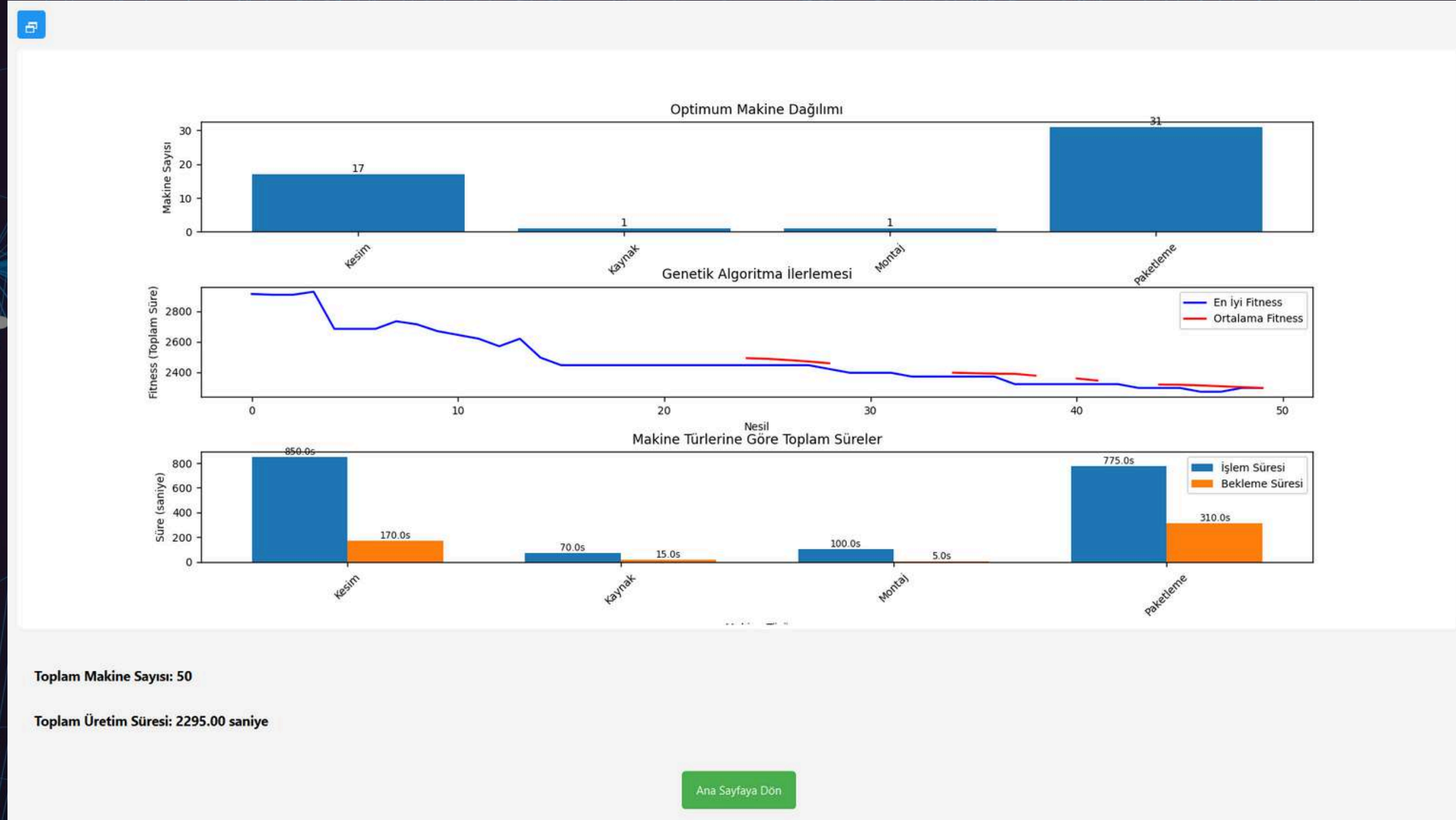
İşlem Süresi (saniye): 25

Bekleme Süresi (saniye): 10

Optimize Et

Her birinin işlem ve bekleme süresi yazılır

Uygulama



İşlem sonucunda Genetik Algoritma Yapay Zekası çalışır ve süre konusunda en iyi verim elde edilir.

AI

TEŞEKKÜRLER