

Rapor

Ödevde girdi olarak verilen resmin, bir çemberin koordinatları ve sezgisel bir algoritma kullanılarak benzer bir şekilde çizilmesi beklenmektedir. Mevcut ödevde algoritma olarak Genetik Algoritma kullanılmıştır. Genetik algoritma çalıştırılırken ilk jenerasyon rastgele bir şekilde yaratılmıştır. Bu jenerasyon kodun ilerleyen kısımlarında genetik algoritmalarla kullanılan crossover, mutasyon ve fitness fonksiyonları ile yeni jenerasyonlar oluşturmada kullanılmıştır. Aşağıdaki ekran görüntüsü ilk rastgele noktalardan oluşan jenerasyonun oluşturulduğu kodu göstermektedir.

```
#ilk jenerasyon icin random kromozomlar uretilir
for i in range(num_of_pop):
    for j in range(num_of_cromosome):
        deger = random.randint(0, 360)
        if (j != 0):
            while (deger == all_generation[i][j - 1]):
                deger = random.randint(0, 360)
        all_generation[i][j] = deger
```

Oluşturulan bu jenerasyon devamında elde edilmiş olan çizgilerin uygunluk puanlarını belirleyen fitness fonksiyonuna verilmektedir. Fitness fonksiyonu popülasyon içerisinde her biri çizimi ifade eden kromozomları ve yine verilmiş olan resmi girdi olarak alır. Aldığı her kromozomu PIL kütüphanesini kullanarak noktaları birleştirecek bir şekilde png uzantılı bir resme çevirir. Devamında oluşturulan resmi ve girdi olarak verilen resmi korelasyon fonksiyonunu kullanarak karşılaştırır ve uygunluk değerini geri döndürür. Bu fonksiyon popülasyon içerisindeki bütün kromozomlar için kullanılır ve çıktılar ayrı bir array içerisinde tutulur. Aşağıda fitness fonksiyonu ve bu fonksiyonu her kromozom için çağrıran fonksiyonun kodları verilmiştir.

```

"""
Bu fonksiyon kromozomların uygunluklarını puanlayan fonksiyondur
Kromozom değerlerine göre solution.png adlı bir dosyaya trigo fonksiyonlarından yararlanarak çizim yapar
Devamında hem input dosyasını hem de solution.png dosyasını opencv yardımıyla okuyup numpy aracılığı ile array haline getirilir
Elde edilen arraylar korelasyon fonksiyonuna tabii tutulur ve elde edilen değer geri dondurulur
"""

def calculate_fitness(kromozom, num_points, input_img):
    resim2 = cv2.imread(input_img, cv2.IMREAD_GRAYSCALE)

    width, height = resim2.shape

    #daireyi ve noktaları ayarlar
    circle_radius = np.sqrt(width ** 2 + height ** 2) / 2
    center_x = width / 2
    center_y = height / 2
    angles = np.linspace(0, 2 * np.pi, num_points, endpoint=False)
    points_on_circle = [(center_x + circle_radius * np.cos(angle), center_y + circle_radius * np.sin(angle)) for angle in angles]

    solution_img = Image.new('L', (width, height), color=255)

    #noktaları birleştiren (variable) points_on_circle: list[tuple[Any, Any]]
    for i in range(1):
        start_point = points_on_circle[i]
        end_point = points_on_circle[kromozom[i + 1] - 1]
        draw = ImageDraw.Draw(solution_img)
        draw.line([start_point, end_point], fill=0, width=1)

    solution_img = solution_img.resize((width, height))
    solution_img.save("try.png")

    solution_array = np.array(solution_img).flatten()

    reference_array = np.array(resim2).flatten()

    korelasyon_katsayisi = np.corrcoef(solution_array, reference_array)[0, 1]
    return korelasyon_katsayisi

#jenerasyonu input olarak alır ve bütün kromozomları sırasıyla fitness fonksiyonuna tabi tutup uygunluk değerlerini array içerisinde dondurur
def fonkhesap(all_generation, points_of_generation, num_of_pop, input_image_path):

    for i in range(num_of_pop):
        benzerlik_orani = calculate_fitness(all_generation[i], 360, input_image_path)
        points_of_generation[i] = benzerlik_orani

```

Elde edilen bu uygunluk değerleri crossover için daha uygun olan değerlerin seçilme miktarı daha yüksek olacak şekilde seçim yapan rastgele_sec fonksiyonuna girdi olarak verilmektedir. Bu fonksiyon seçim için sıralama seçim yöntemini kullanmaktadır. Aşağıda seçim fonksiyonunun kodunu gösteren ekran fotoğrafı verilmiştir.

```

"""
Bu fonksiyon sıralama seçiminin uygunluk değeri yüksek olanlar için daha yüksek ihtimalle seçim yapan fonksiyondur
Fonksiyonda sıralama seçiminde olduğu gibi uygunluk değerlerini sıralar ve ihtimalleri indis değerleri /toplam değerler olarak belirler
Devamında random fonksiyonundan uygun bir şekilde yararlanabilmek için bu ihtimalerin kumulatif toplamlarından yeni bir dizi üretir
Boylece random fonksiyonunun üreteceği değerler kumulatif toplam array'i üzerinde uygun raliga uygun ihtimalerde düşecektir
"""

def rastgele_sec(dizi,num_of_pop):

    sorted_array = np.sort(dizi)

    new_fitness_values = np.arange(1,len(sorted_array) + 1)

    probabilities = new_fitness_values / np.sum(new_fitness_values)

    cumulative_sum_probabilities = np.cumsum(probabilities)

    #a number between 0-1
    random_probability = random.random()

    index = 0
    #find the slot where random number hits
    for i in range(len(cumulative_sum_probabilities)):
        if(random_probability >= cumulative_sum_probabilities[i]):
            index = i + 1

    value = sorted_array[index]
    returned_index = np.where(dizi == value)[0][0]

    return returned_index

```

Bu fonksiyondan elde edilen iki index popülasyon içerisinde crossover yapılacak iki kromozomu belirler. Elde edile bu indexe at kromozomlar rastgele bir noktalarından kesilerek crossover işlemine tabi tutulur. Crossover işlemlerini gösteren kodlar aşağıda verilmiştir.

```
#Random değerlerle indis belirler ve ona göre iki kromozomun farklı parçalarını birleştirerek yeni kromozomlar üretir
def crossover(dizi1, dizi2):
    uzunluk = len(dizi1)
    yeni_dizi = np.zeros(uzunluk)

    # Çaprazlama noktasını rastgele seç
    caprazlama_noktası = np.random.randint(1, uzunluk)

    # Eşit olmadığı sürece rastgele seç
    while dizi1[caprazlama_noktası - 1] == dizi2[caprazlama_noktası - 1]:
        caprazlama_noktası = np.random.randint(1, uzunluk)

    # Yeni diziyi oluşturma
    yeni_dizi[:caprazlama_noktası] = dizi1[:caprazlama_noktası]
    yeni_dizi[caprazlama_noktası:] = dizi2[caprazlama_noktası:]

    return yeni_dizi
```

Crossover sonrası elde edilen bütün kromozomlar arasından yine rastgele bir şekilde seçilen kromozomlar üzerinde mutasyon işlemi gerçekleştirilir. Bu işlem parametre olarak verile mutasyon oranına göre kromozomlarda değişiklik ihtimalini belirler. Aşağıda mutasyon işlemini gerçekleştiren kod gösterilmiştir.

```
#verilen mutasyon oranına göre random bir sayı aracılığıyla mutasyon gerçekleştirir
def degisiklik_yap(dizi,mutation_rate):

    for k in range(num_of_pop):
        for l in range(num_of_cromosome):
            if np.random.rand() < mutation_rate:
                # Değiştirilecek yeni değeri 0 ile 360 arasında rastgele seçelim
                yeni_deger = random.randint(0, 360)

                #uretilen noktanın aynı nokta olmadığından emin olur
                if ((l != 0 and yeni_deger != dizi[k][l - 1]) or l == 0):
                    dizi[k][l] = yeni_deger

    return dizi
```

Polar grafik çizdirme ve ana işlemlerin gerçekleştiği kod parçaları aşağıda gösterilmiştir.

```
#ilk jenerasyon icin random kromozomlar uretilir
for i in range(num_of_pop):
    for j in range(num_of_chromosomes):
        deger = random.randint(0, 360)
        if (j != 0):
            while (deger == all_generation[i][j - 1]):
                deger = random.randint(0, 360)
        all_generation[i][j] = deger

points_of_generation = np.zeros(num_of_pop)
iterasyon = 1

#genetik algoritma belirlenen iterasyon sayısında calistirilir
while (num_of < total_iteration_number):
    #jenerasyon (variable) all_generation: ndarray[Any, dtype[int_]] fonksiyon
    fonkhesap(all_generation, points_of_generation, num_of_pop, input_image_path)
    num_of = num_of + 1
    tmp_all_generation = all_generation

    #elde edilen uygunluk degerleri üzerinden crossover icin rastgele secim yapan fonksiyon
    #uygunluk degeri yüksekse secilme ihtimali daha yüksek
    for i in range(num_of_pop):
        ilk = rastgele_sec(points_of_generation,num_of_pop)
        iki = rastgele_sec(points_of_generation,num_of_pop)
        yedekyedek = crossover(all_generation[ilk], all_generation[iki])
        tmp_all_generation[i] = yedekyedek

    all_generation = tmp_all_generation
    #belirli oranda mutasyonu uygulayan fonksiyon
    all_generation = degisiklik_yap(all_generation,mutation_rate)

    #değerlerin bastırılması
    max_index = np.argmax(points_of_generation) # argmax en büyük elemanın indexini dondurur
    print(iterasyon)
    print("max index = " + str(max_index))
    print("değer " + str(points_of_generation[max_index]))
    mean = np.mean(points_of_generation)

    mean_s.append(mean)#baslangic burada
    max_values.append(points_of_generation[max_index])

    print("mean" + str(mean))
    iterasyon = iterasyon + 1
    # print(all_generation[max_index])
    if (num_of % 10 == 0):
        polar_grafik_ciz(all_generation[max_index], input_image_path, num_of)
```

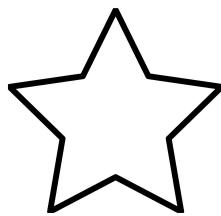
```
fig, axs = plt.subplots(2, figsize=(8, 6)) # 2 satır, 1 sütunlu bir alt grafik düzeni

# Mean grafiğini oluştur
axs[0].plot(mean_s, color='blue', marker='o', linestyle='-', label='Mean Values')
axs[0].set_xlabel('Index') # X eksenini etiketi
axs[0].set_ylabel('Mean Values') # Y eksenini etiketi
axs[0].set_title('Mean Values by Index') # Grafiğin başlığı
axs[0].legend() # Gösterilen verilerin açıklamalarını ekle
axs[0].grid(True) # Izgarayı göster

# Max values grafiğini oluştur
axs[1].plot(max_values, color='red', marker='s', linestyle='--', label='Max Values')
axs[1].set_xlabel('Index') # X eksenini etiketi
axs[1].set_ylabel('Max Values') # Y eksenini etiketi
axs[1].set_title('Max Values by Index') # Grafiğin başlığı
axs[1].legend() # Gösterilen verilerin açıklamalarını ekle
axs[1].grid(True) # Izgarayı göster

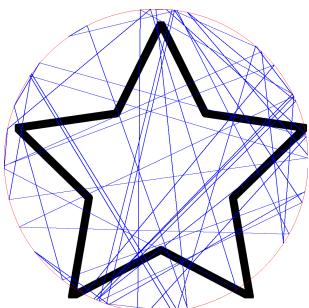
plt.tight_layout() # Grafikler arasındaki boşluğu ayarla
plt.show() # Grafikleri göster
```

Yıldız

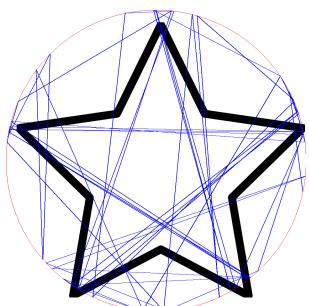


Parametreler

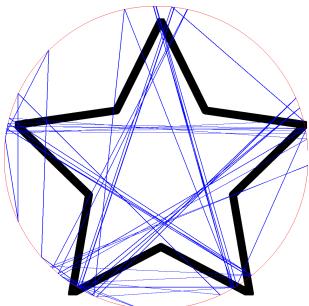
Popülasyon miktarı: 600
Nokta miktarı: 50
Mutasyon oranı: 0.01
Jenerasyon sayısı: 300



Jenerasyon 10



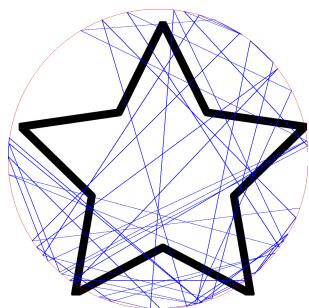
Jenerasyon 150



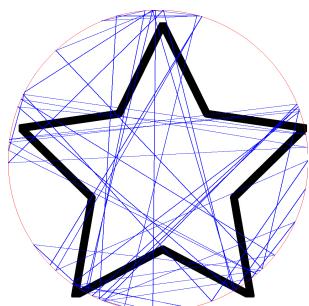
Jenerasyon 300

Parametreler

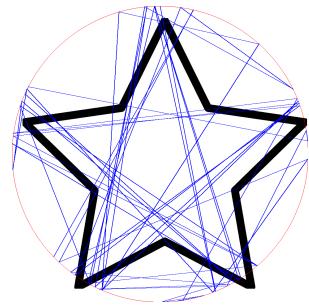
Popülasyon miktarı: 600
Nokta Miktarı: 50
Mutasyon Oranı: 0.03
Jenerasyon Sayısı: 300



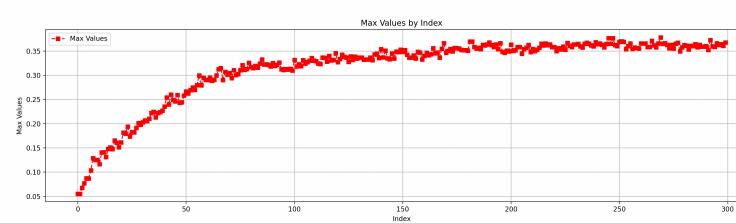
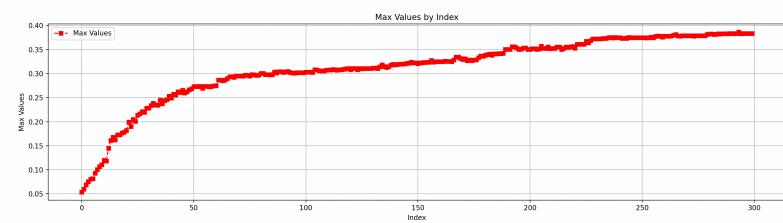
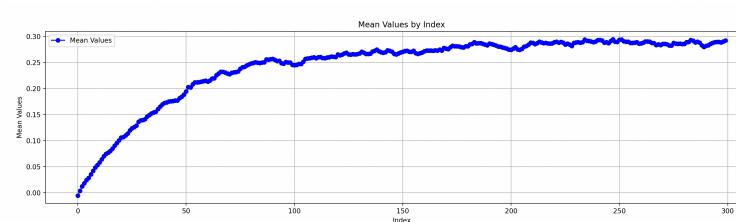
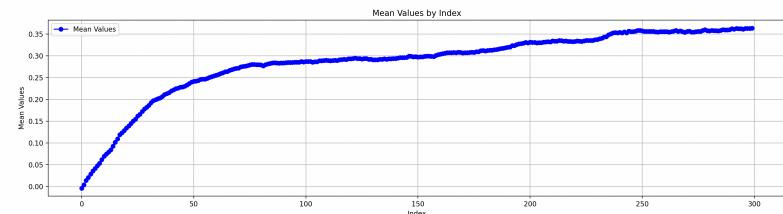
Jenerasyon 10



Jenerasyon 150



Jenerasyon 300



Yorum:

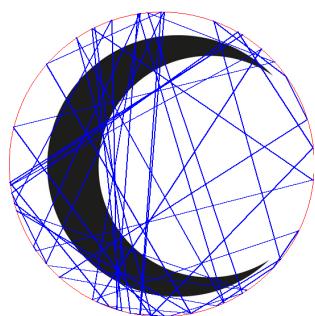
Yıldız şecline benzemeye çalışan algoritmamızda diğer değerler sabit tutularak mutasyon oranımız 0.01 den 0.03 e artırılmış ve etki olarak; bir nebze daha başarısız bir resim(benzerlik oranları 0.01 için:0.36 0.03 için 0.29),0.03 mutasyonlu algoritmada max_değerlerimizdeki dengesizlik,peak değerde kalma konusunda da bir sıkıntı ,mean değerlerin yükselememesi durumu gözlenmiştir.Bu resim için mutasyon miktarını az tutmak daha mantıklı olacaktır.

Hilal



Parametreler

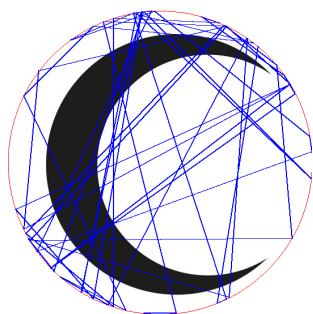
Popülasyon miktarı: 600
Nokta miktarı: 50
Mutasyon oranı: 0.01
Jenerasyon sayısı: 300



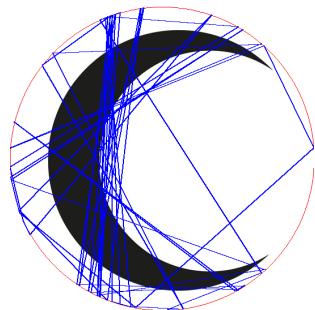
Jenerasyon 10

Parametreler

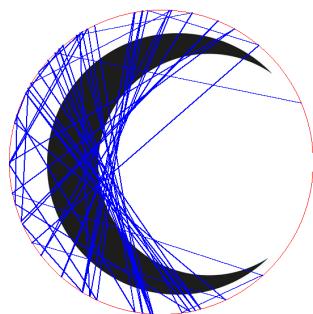
Popülasyon miktarı: 600
Nokta miktarı: 50
Mutasyon oranı: 0.03
Jenerasyon sayısı: 300



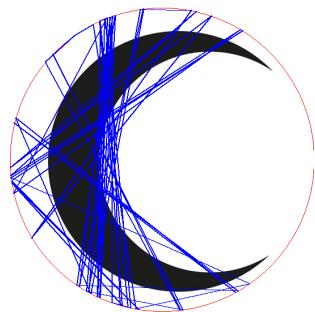
Jenerasyon 10



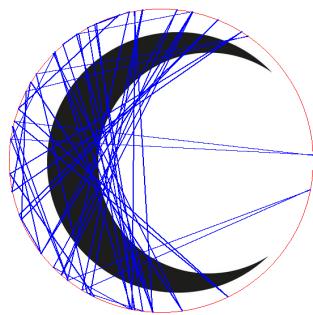
Jenerasyon 150

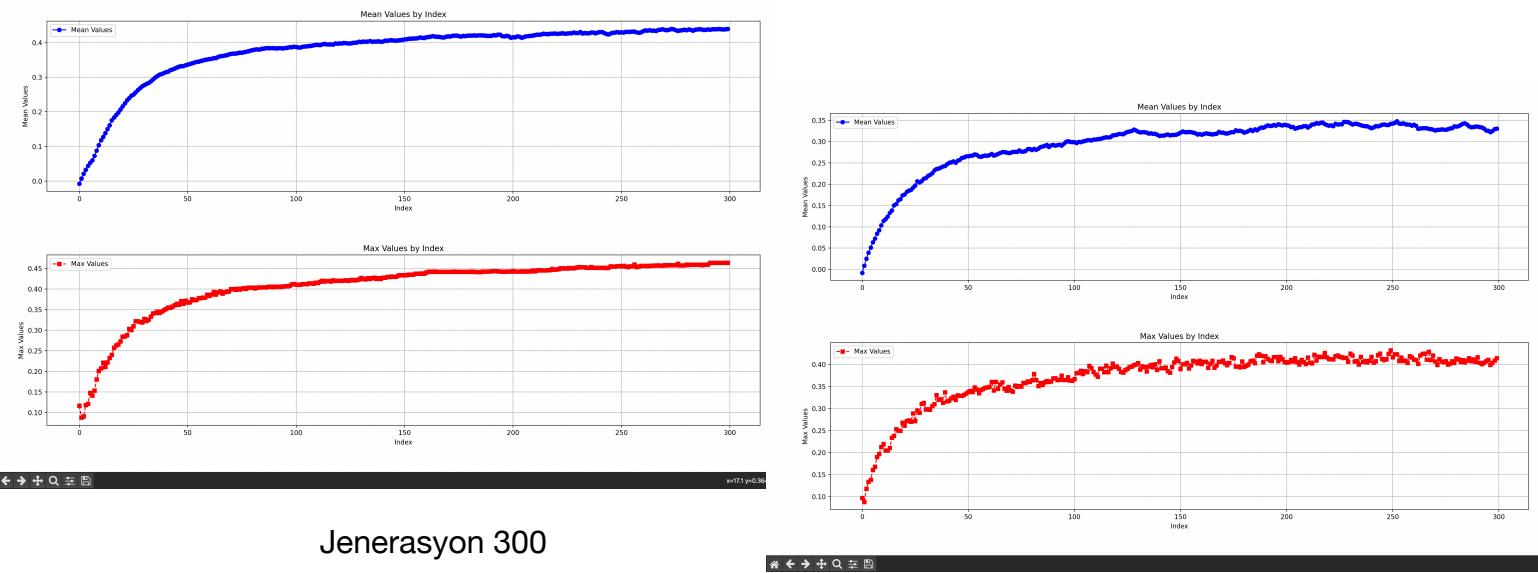


Jenerasyon 150



Jenerasyon 300



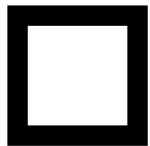


Jenerasyon 300

Yorum:

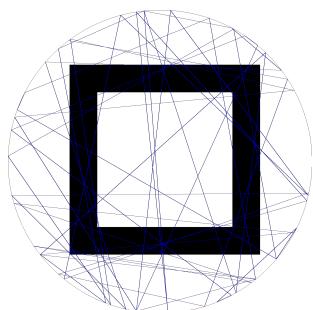
Hilal şeklinde benzemeye çalışan algoritmamızda diğer değerler sabit tutularak mutasyon oranımız 0.01 den 0.03 e artırılmış ve etki olarak;(benzerlik oranları 0.01 için:0.42 0.03 için 0.32) olmasına rağmen insan gözüyle çok daha başarılı (lokal maksimumdan çıktığı için),0.04 mutasyonlu algoritmada max_değerlerimizdeki dengesizlik,peak değerde kalma konusunda da bir sıkıntı ,mean değerlerin yükselememesi durumları hala gözlenmiştir.Bu resim için mutasyon miktarını biraz fazla tutmak görsel açıdan daha mantıklı olacaktır.

Kare

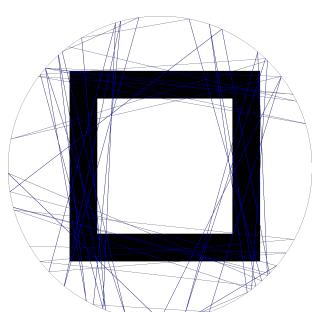


Parametreler

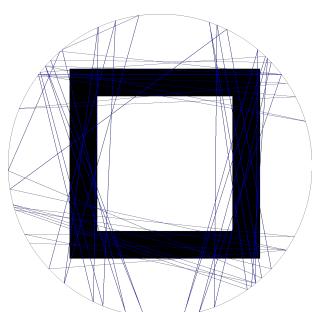
Popülasyon miktarı: 600
Nokta miktarı: 50
Mutasyon oranı: 0.01
Jenerasyon sayısı: 300



Jenerasyon 10



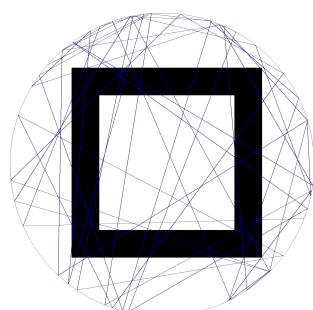
Jenerasyon 150



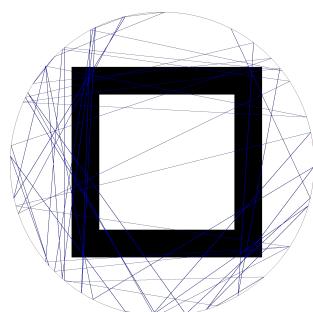
Jenerasyon 300

Parametreler

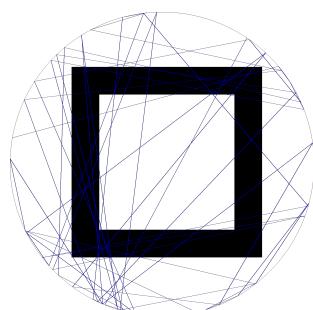
Popülasyon miktarı: 600
Nokta miktarı: 50
Mutasyon oranı: 0.04
Jenerasyon sayısı: 300

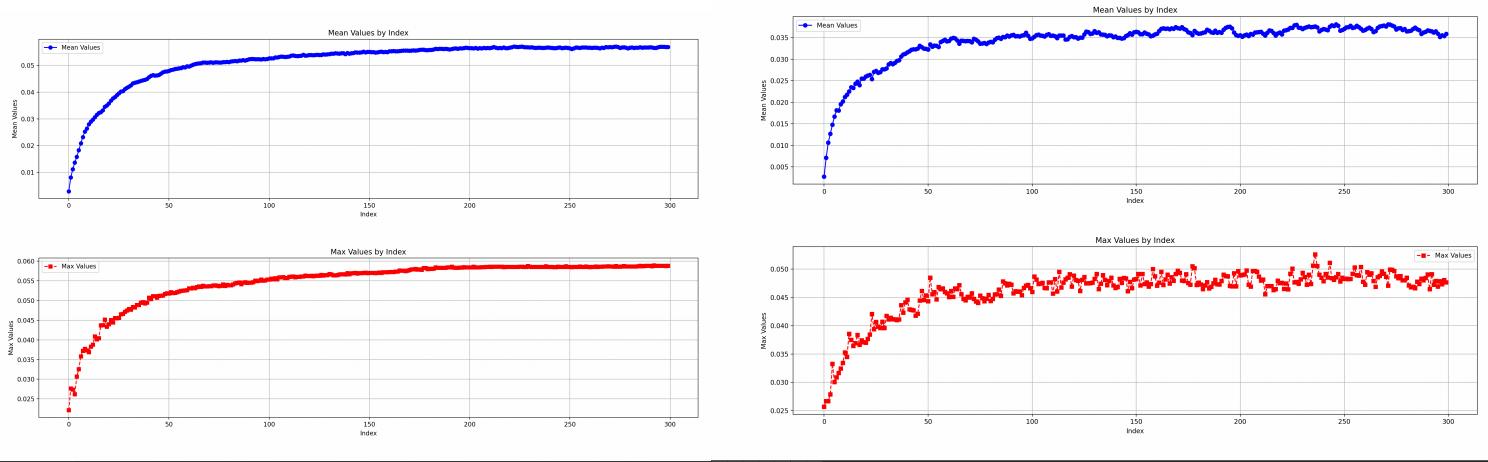


Jenerasyon 10



Jenerasyon 150





Jenerasyon 300

Yorum:

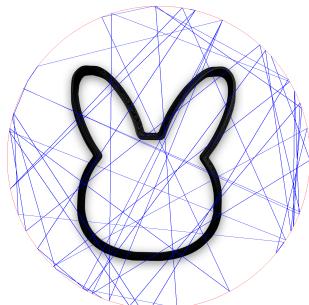
Kare şekline benzemeye çalışan algoritmamızda diğer değerler sabit tutularak mutasyon oranımız 0.01 den 0.04 e artırılmış ve etki olarak; çok daha başarısız bir resim(benzerlik oranları 0.01 için:0.36 0.04 için 0.055) olmasına rağmen insan gözüyle çok daha başarısız,0.04 mutasyonlu algoritmada max_değerlerimizdeki dengesizlik,peak değerde kalma konusunda da bir sıkıntı ,mean değerlerin yükselememesi durumu gözlenmiştir.Bu resim için mutasyon miktarını az tutmak çok daha mantıklı olacaktır.

Tavşan



Parametreler

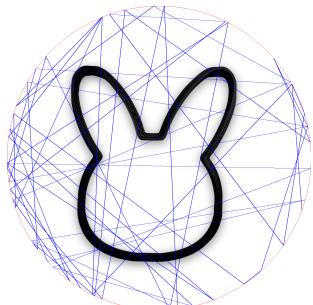
Popülasyon miktarı: 600
Nokta miktarı: 50
Mutasyon oranı: 0.01
Jenerasyon sayısı: 300



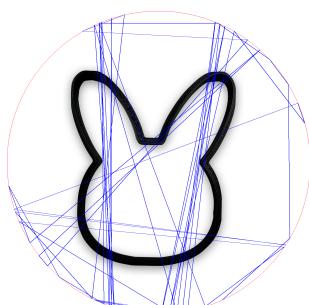
Jenerasyon 10

Parametreler

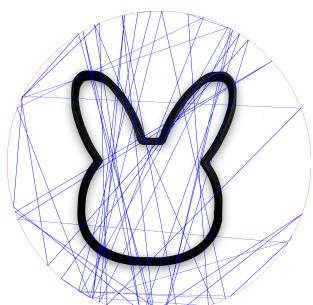
Popülasyon miktarı: 600
Nokta miktarı: 50
Mutasyon oranı: 0.01
Jenerasyon sayısı: 300



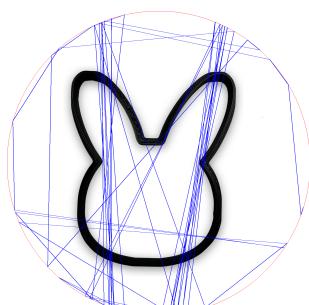
Jenerasyon 10



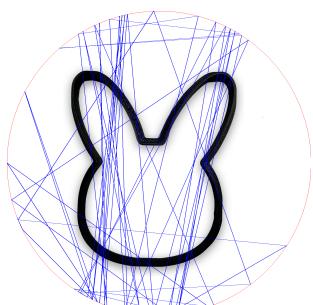
Jenerasyon 150



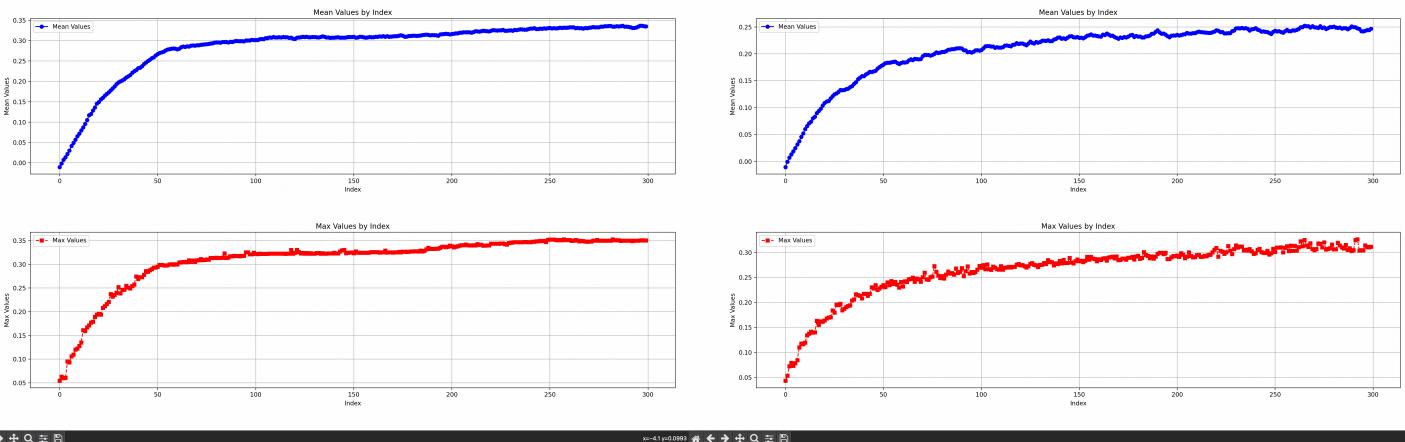
Jenerasyon 150



Jenerasyon 300



Jenerasyon 300



Yorum:

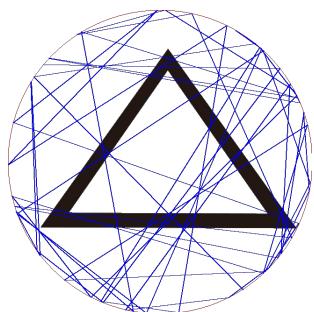
Tavşana benzemeye çalışan algoritmamızda diğer değerler sabit tutularak mutasyon oranımız 0.01 den 0.03 e arttırılmış ve etki olarak farkedilebilir derecede daha başarısız bir resim(benzerlik oranları 0.01 için:0.34 0.06 için 0.24),0.06mutasyonlu algoritmada max_değerlerimizdeki dengesizlik,peak değerde kalma konusunda da bir sıkıntı ,mean değerlerin yükeselememesi durumu gözlenmiştir.Bu resim için mutasyon miktarını az tutmak kesinlikle daha mantıklı olacaktır.

Üçgen



Parametreler

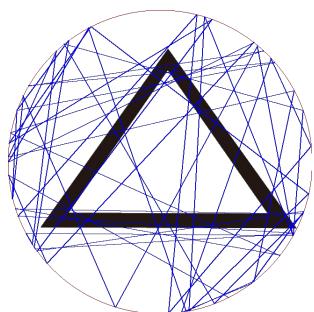
Popülasyon miktarı: 600
Nokta miktarı: 50
Mutasyon oranı: 0.05
Jenerasyon sayısı: 300



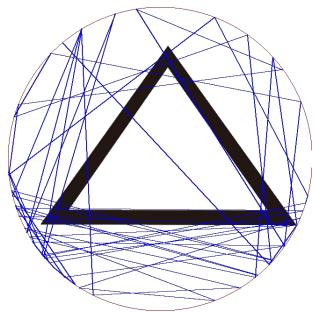
Jenerasyon 10

Parametreler

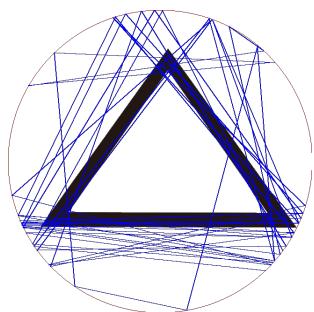
Popülasyon miktarı: 600
Nokta miktarı: 50
Mutasyon oranı: 0.01
Jenerasyon sayısı: 300



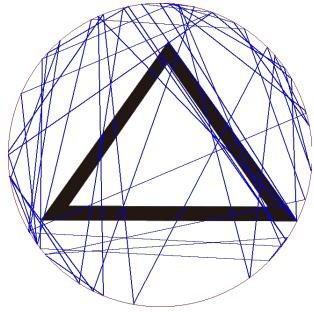
Jenerasyon 10



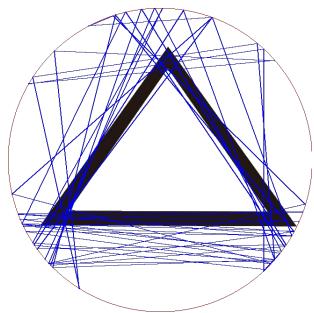
Jenerasyon 150



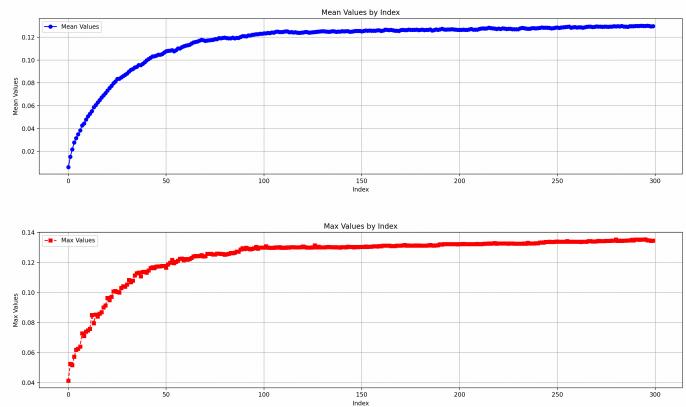
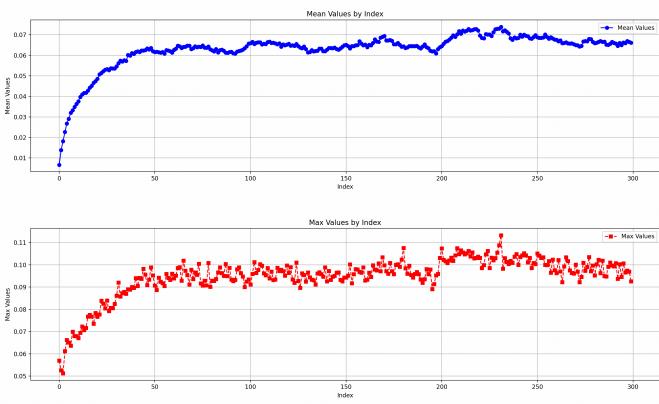
Jenerasyon 150



Jenerasyon 300



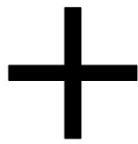
Jenerasyon 300



Yorum:

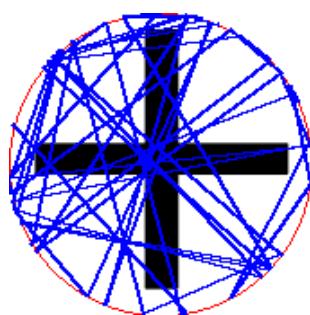
Üçgene benzemeye çalışan algoritmamızda diğer değerler sabit tutularak mutasyon oranımız 0.05 den 0.01 e düşürülmüştür ve etki olarak; farkedilebilir derecede daha başarılı bir resim(benzerlik oranları 0.05 için:0.06 0.06 için 0.013), 0.05 mutasyonlu algoritmda max_değerlerimizdeki denge,peak değerde kalma konusunda bir başarı ,mean değerlerin satabil olması durumu gözlenmiştir.Bu resim için mutasyon miktarını az tutmak kesinlikle daha mantıklı olacaktır.

Artı

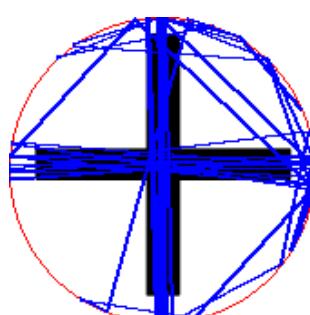


Parametreler

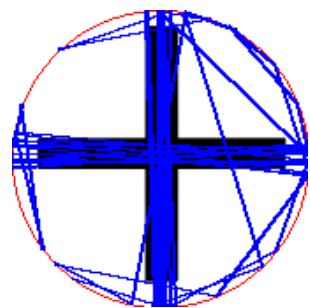
Popülasyon miktarı: 200
Nokta miktarı: 50
Mutasyon oranı: 0.01
Jenerasyon sayısı: 300



Jenerasyon 10



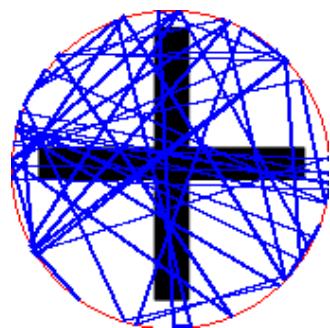
Jenerasyon 150



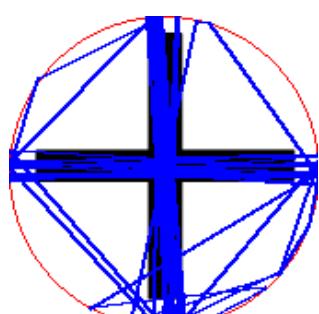
Jenerasyon 300

Parametreler

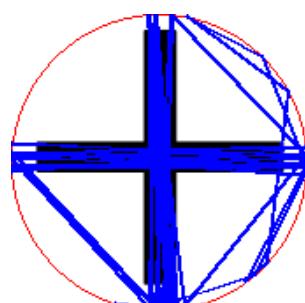
Popülasyon miktarı: 500
Nokta miktarı: 50
Mutasyon oranı: 0.01
Jenerasyon sayısı: 300



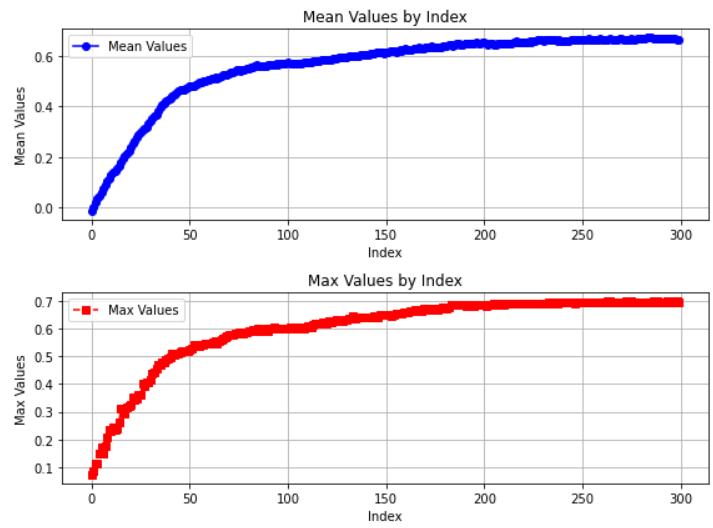
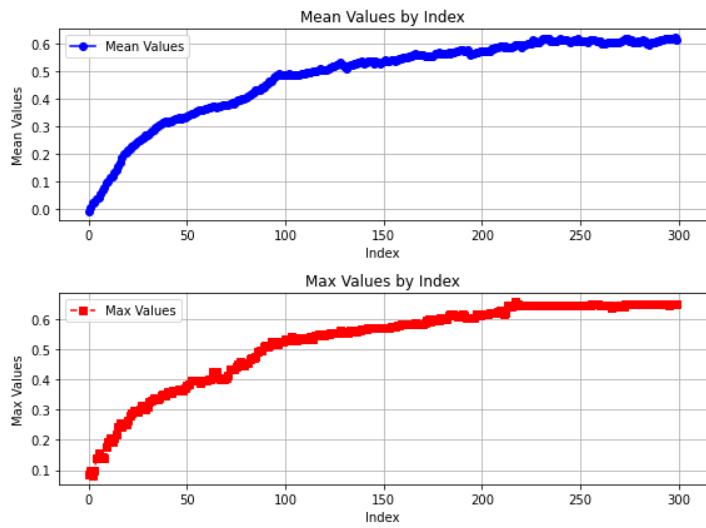
Jenerasyon 10



Jenerasyon 150



Jenerasyon 300



Yorum:

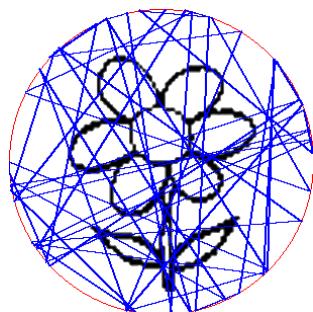
Artı resmine benzetilmeye çalışılan algoritmada, diğer değerler sabit tutularak populasyon sayısı 500 den 200 e azaltılmış ve etki olarak; görece daha başarısız bir resim ,azalan çalışma zamanı, daha erken peak değere ulaşmak olmuştur.Bu resim özelinde populasyon sayısı olarak 200 kabul etmek daha avantajlı bir durum olarak görülmektedir.

Çiçek



Parametreler

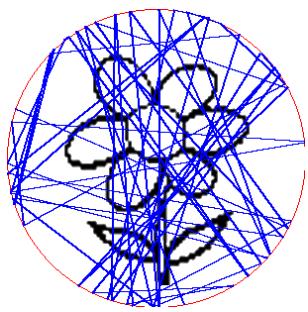
Popülasyon miktarı: 100
Nokta miktarı: 50
Mutasyon oranı: 0.01
Jenerasyon sayısı: 300



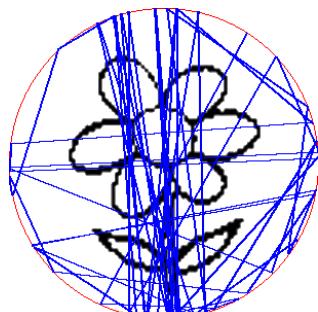
Jenerasyon 10

Parametreler

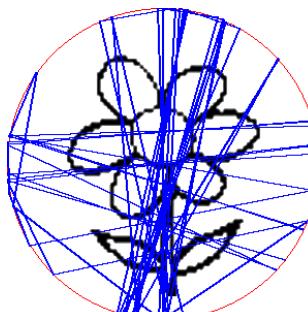
Popülasyon miktarı: 500
Nokta miktarı: 50
Mutasyon oranı: 0.01
Jenerasyon sayısı: 300



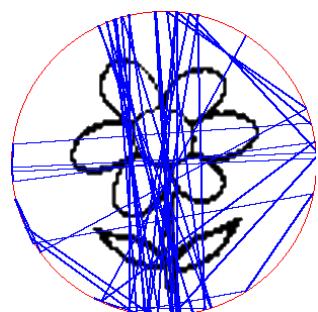
Jenerasyon 10



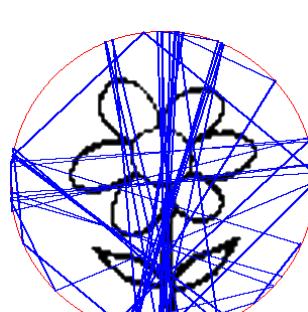
Jenerasyon 150



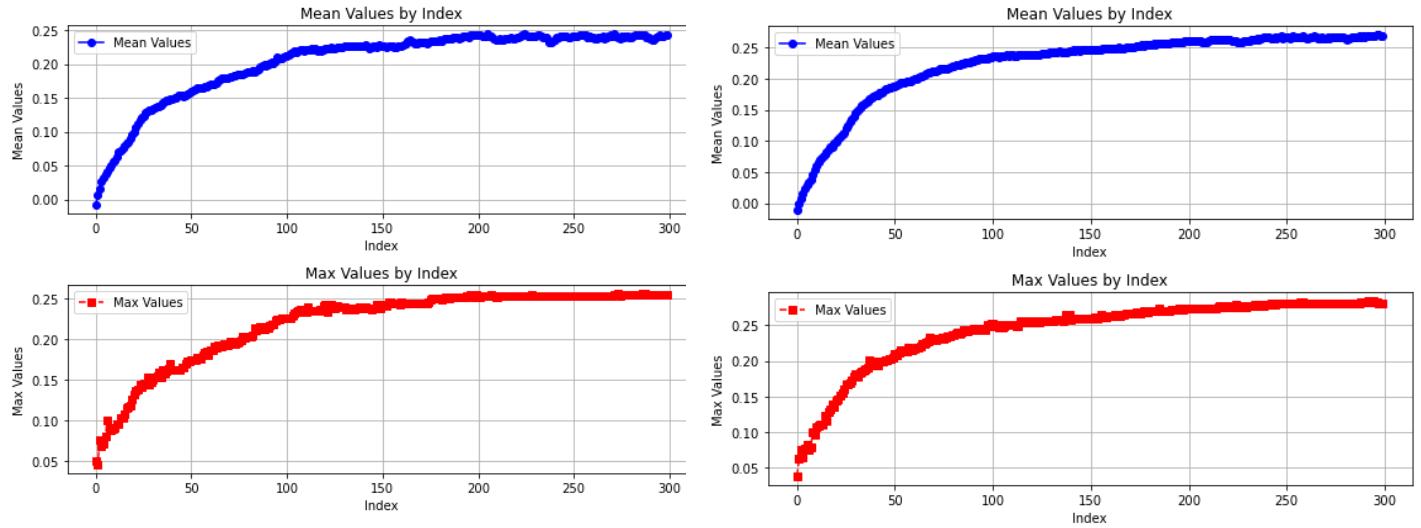
Jenerasyon 150



Jenerasyon 300



Jenerasyon 300



Yorum:

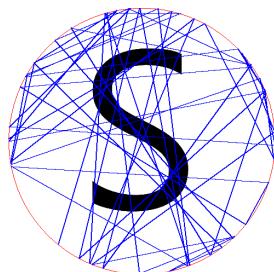
Artı resmine benzetilmeye çalışılan algoritmada, diğer değerler sabit tutularak populasyon sayısı 500 den 200 e azaltılmış ve etki olarak; görece daha başarısız bir resim ,azalan çalışma zamanı,daha erken peak değere ulaşmak olmuştur.Bu resim özelinde populasyon sayısı olarak 200 kabul etmek daha avantajlı bir durum olarak görülmektedir.

S Harfi

S

Parametreler

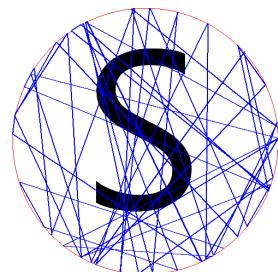
Popülasyon miktarı: 200
Nokta miktarı: 50
Mutasyon oranı: 0.01
Jenerasyon sayısı: 300



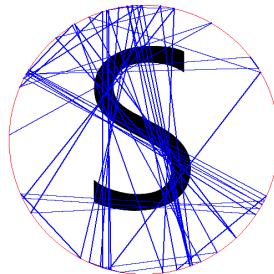
Jenerasyon 10

Parametreler

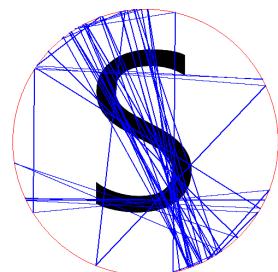
Popülasyon miktarı: 500
Nokta miktarı: 50
Mutasyon oranı: 0.01
Jenerasyon sayısı: 300



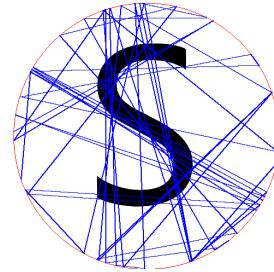
Jenerasyon 10



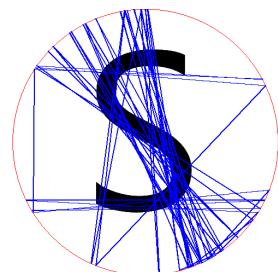
Jenerasyon 150



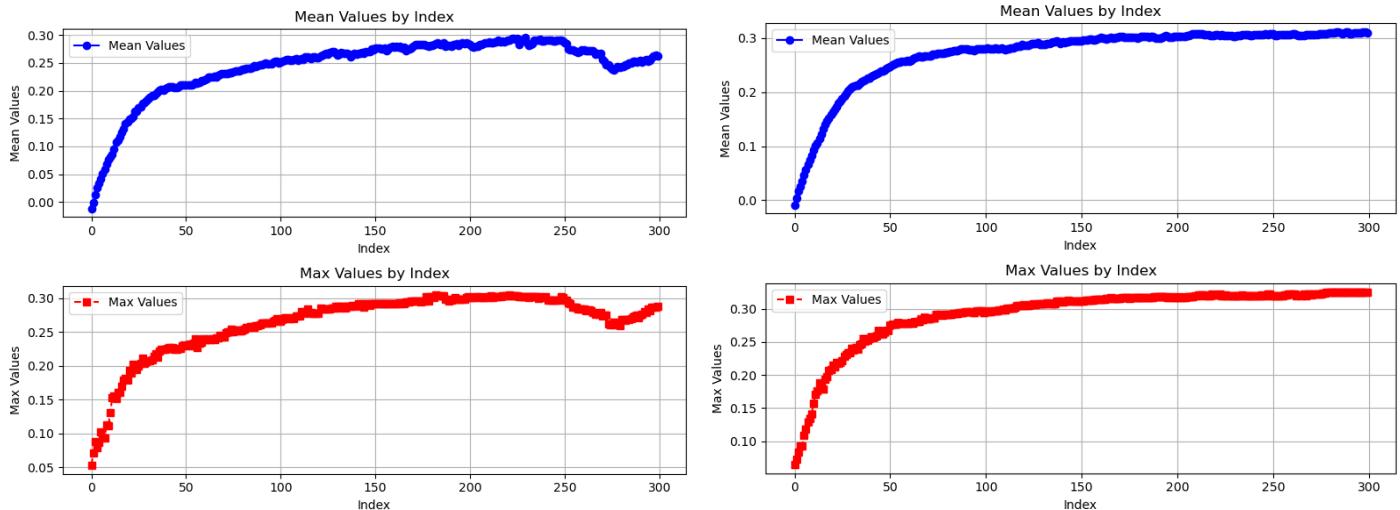
Jenerasyon 150



Jenerasyon 300



Jenerasyon 300



Yorum:

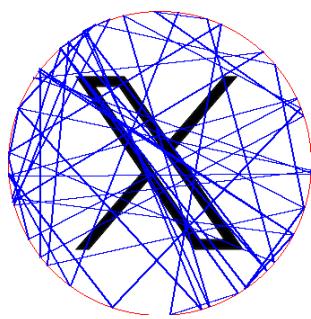
Artı resmine benzetilmeye
çalışılan algoritmada, diğer
değerler sabit tutularak
populasyon sayısı 500 den 200 e
azaltılmış ve etki olarak; görece
daha başarısız bir resim ,azalan
çalışma zamanı, daha erken peak
değere ulaşmak olmuştur.Bu
resim özelinde populasyon sayısı
olarak 200 kabul etmek daha
avantajlı bir durum olarak
görülmektedir.

Twitter X

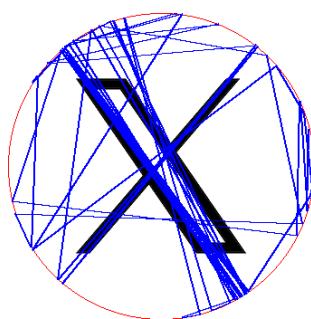


Parametreler

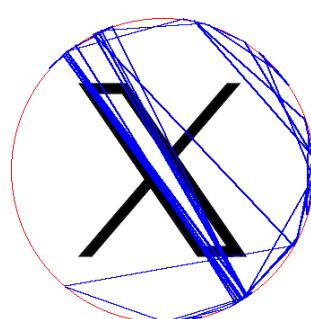
Popülasyon miktarı: 200
Nokta miktarı: 50
Mutasyon oranı: 0.01
Jenerasyon sayısı: 300



Jenerasyon 10



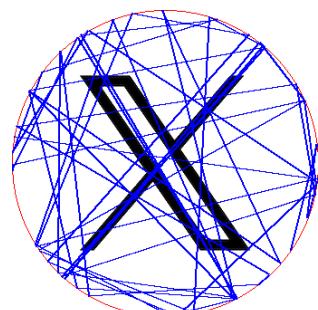
Jenerasyon 150



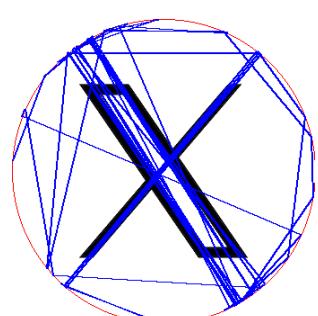
Jenerasyon 300

Parametreler

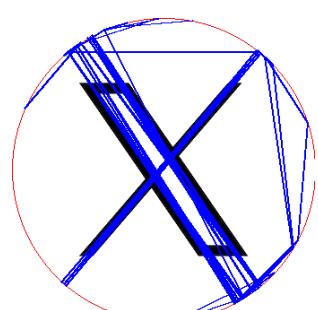
Popülasyon miktarı: 500
Nokta miktarı: 50
Mutasyon oranı: 0.01
Jenerasyon sayısı: 300



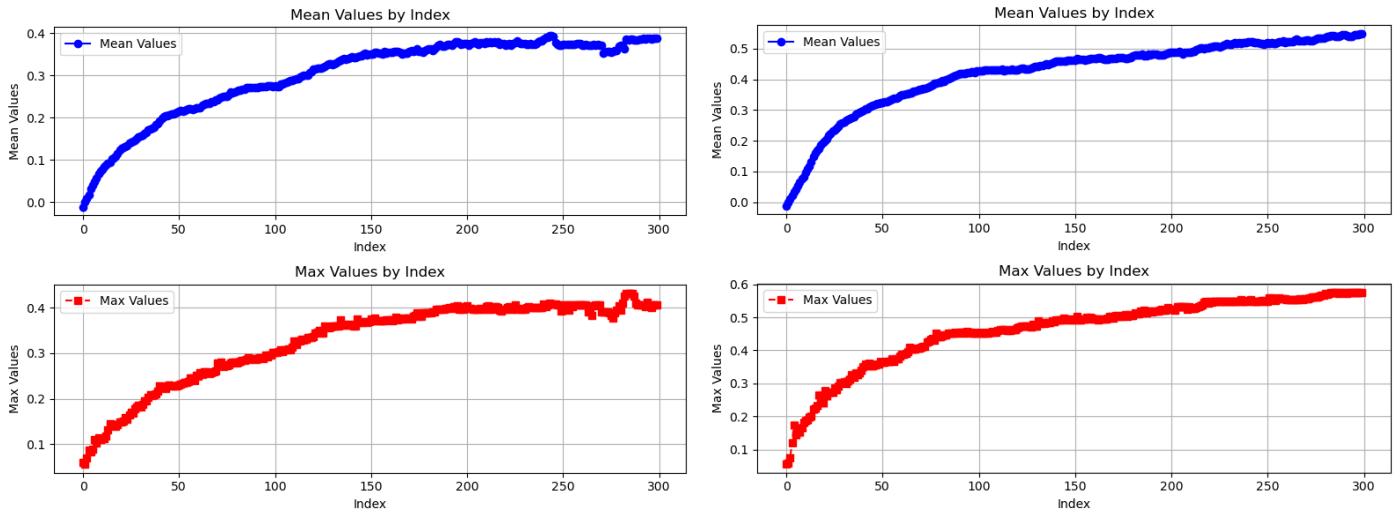
Jenerasyon 10



Jenerasyon 150



Jenerasyon 300



Yorum:

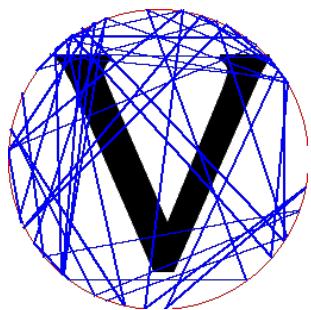
Artı resmine benzetilmeye
çalışılan algoritmada, diğer
değerler sabit tutularak
populasyon sayısı 500 den 200 e
azaltılmış ve etki olarak; görece
daha başarısız bir resim ,azalan
çalışma zamanı,daha erken peak
değere ulaşmak olmuştur.Bu
resim özelinde populasyon sayısı
olarak 200 kabul etmek daha
avantajlı bir durum olarak
görülmektedir.

V

V

Parametreler

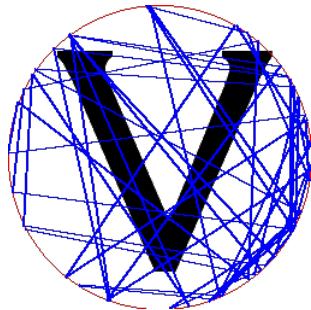
Popülasyon miktarı: 600
Nokta miktarı: 50
Mutasyon oranı: 0.01
Jenerasyon sayısı: 300



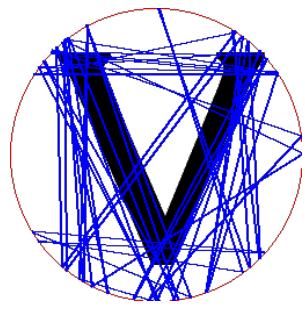
Jenerasyon 10

Parametreler

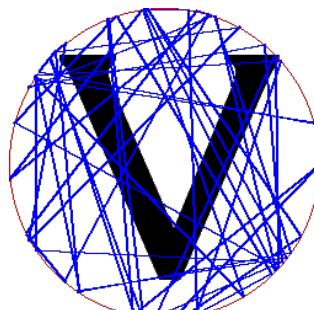
Popülasyon miktarı: 600
Nokta miktarı: 50
Mutasyon oranı: 0.05
Jenerasyon sayısı: 300



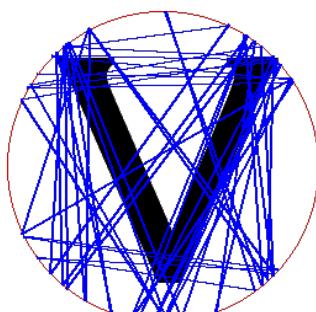
Jenerasyon 10



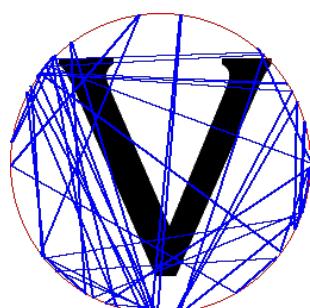
Jenerasyon 150



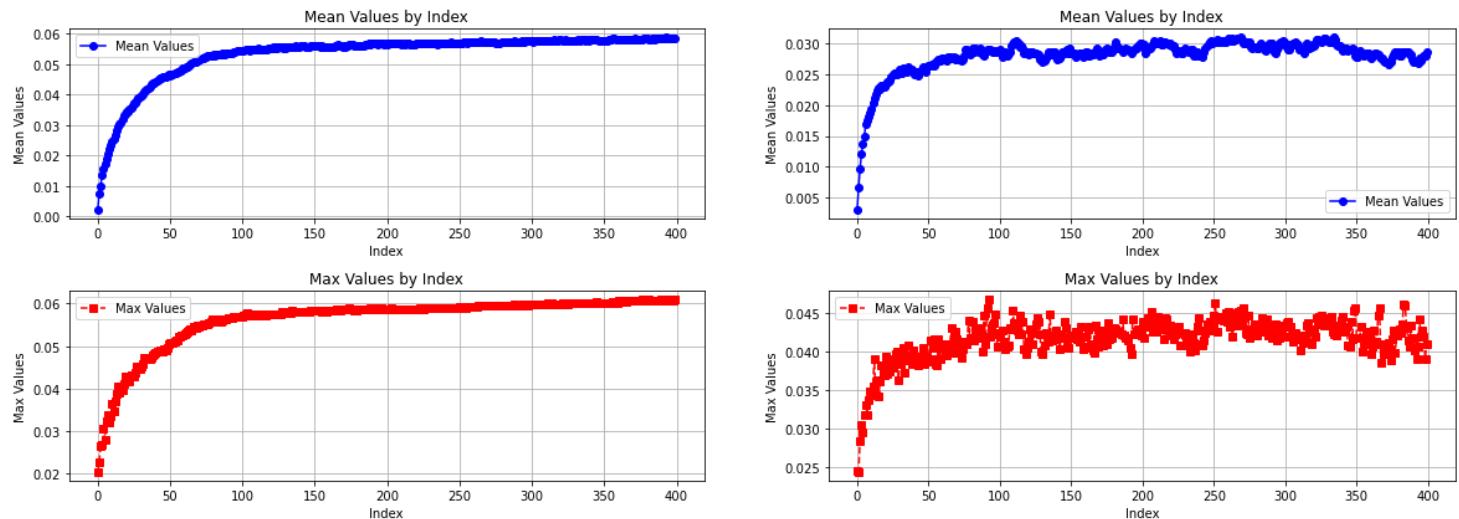
Jenerasyon 150



Jenerasyon 300



Jenerasyon 300



Yorum:

Artı resmine benzetilmeye çalışılan algoritmada, diğer değerler sabit tutularak populasyon sayısı 500 den 200 e azaltılmış ve etki olarak; görece daha başarısız bir resim ,azalan çalışma zamanı, daha erken peak değere ulaşmak olmuştur.Bu resim özelinde populasyon sayısı olarak 200 kabul etmek daha avantajlı bir durum olarak görülmektedir.

Video Link:

<https://youtu.be/P1mQFj6BPPk>

Hazırlayanlar:

İsim: Metehan Türkmen
Numara: 20011048

İsim: Said Enes Subaşı
Numara: 20011060