CENG 415 Evrimsel Hesaplama

Bölüm 7: DEAP(**D**istributed **E**volutionary **A**lgorithms in **P**ython)

Şevket Umut Çakır

Pamukkale Üniversitesi

7 Aralık 2022

Anahat

DEAP[6]

- Python programlama dilinde evrimsel algoritmalar geliştirmek için kullanılan bir pakettir.
- Kullanıcıyı ön tanımlı türleri kullanmaya zorlamaz, kullanıcılar kendi türlerini ve temsil biçimlerini kullanabilir.
- Paralel hesaplamayı mümkün kılar
  - SCOOP(Scalable Concurrent Operations in Python) [9]
  - Python multiprocessing: Tek bilgisayar için, kullanımı kolay
- İçerisinde bir çok yöntem ve algoritmayı barındırır.
- Dokümantasyon: `https://deap.readthedocs.io/en/master/`
- Proje: `https://github.com/deap/deap`

Türler

- Problem için uygun bir tür(temsil biçimi) düşünülmelidir.
- `creator` modülü kullanılarak yeni türler oluşturulur.

Tür Oluşturma Örneği

```
from deap import base, creator # Modül yükleme
creator.create("FitnessMin", base.Fitness, weights=(-1.0,)) #
↪  Uygunluk fonksiyonu türü
creator.create("Individual", list, fitness=creator.FitnessMin) #
↪  Birey türü
```

- `creator.create` parametreleri:
  - `name`: Oluşturulacak sınıf adı
  - `base`: Türetilecek temel sınıf
  - `attribute`: Başlatma için bir veya daha fazla parametre, opsiyonel

Başlatma

- Türler oluşturulduktan sonra, içleri genellikle rastgele olarak doldurulur
- `Toolbox` başlatıcılar da dahil olmak üzere her türdeki araçlar için bir kapsayıcıdır

Başlatma Örneği

```
import random # Rastgele sayılar için modül
from deap import tools
IND_SIZE = 10 # Birey boyutu
toolbox = base.Toolbox() # Alet çantası
toolbox.register("attribute", random.random)
toolbox.register("individual", tools.initRepeat, creator.Individual,
↪  toolbox.attribute, n=IND_SIZE)
toolbox.register("population", tools.initRepeat, list, toolbox.individual)
```

Başlatma

- Birey ve popülasyon fonksiyonları alet çantası içine kaydedilmiştir ve çağrılarak başlatılabilir

```
>>> toolbox.individual()
[0.382355593915220 56, 0.033615995747103944, 0.1611679554061327,
↪  0.07639011393012884, 0.9904979982116697, 0.5480334291896338,
↪  0.5088627952230499, 0.07133440740580743, 0.3871698650174773,
↪  0.544516420016302]
>>> toolbox.population()
Traceback (most recent call last):
        File "<stdin>", line 1, in <module>
TypeError: initRepeat() missing 1 required positional argument: 'n'
>>> toolbox.population(n=2)
[[0.325867463636038, 0.8839180611552174, 0.5885587707306784,
↪  0.3560486059915860 5, 0.3459722152517330 5, 0.05773563136204585,
↪  0.8545315167347038, 0.15761771988141393, 0.5692152757317386,
↪  0.618855412978427], [0.9837589186798829, 0.32989768350002,
↪  0.012734059911934215, 0.5745360353235569, 0.8433224326221568,
↪  0.2170770676659679, 0.37721625831910854, 0.1723943517940233,
↪  0.6484624548917144, 0.455178754871677 9]]
```

Başlatma

`tools.initRepeat` Fonksiyonu

- `tools.initRepeat`: Fonksiyonu(`func`) n defa çağırır ve sonucu kapsayıcı(container) türünden döndürür.
- Parametreleri:
  - `container`: Fonksiyondan dönen değerleri saklayacak kapsayıcı türü
  - `func`: Çağrılacak fonksiyon
  - `n`: Çağırma adeti

```
>>> import random
>>> from deap import tools
>>> tools.initRepeat(list, random.random, 3)
[0.7215534457649535, 0.7362641348767816, 0.2838234428460863]
```

Başlatma

`tools.initIterate` Fonksiyonu

- `tools.initIterate`: Kapsayıcı(container) ve üreteç fonksiyonu parametrelerini alan ve kapsayıcıyı üretilen değerlerle dolduran fonksiyon.
- Parametreleri:
  - `container`: Fonksiyondan dönen değerleri saklayacak kapsayıcı türü
  - `generator`: Yinelenebilir(Iterable) bir tür(liste, tuple, vb.) döndüren fonksiyon, bu türün içeriği kapsayıcıyı doldurur.

```
>>> import random
>>> from functools import partial
>>> from deap import tools
>>> gen_idx = partial(random.sample, range(10), 10) #
↪  initIterate kapsayıcı ve tek bir fonksiyonu parametre
↪  olarak aldığından
>>> tools.initIterate(list, gen_idx)
[1, 6, 2, 8, 3, 4, 0, 9, 7, 5]
```

Operatörler

- Mutasyon, çaprazlama ve seçme gibi operatörler mevcuttur.
- Bazıları `tools` modülünde gerçekleştirilmiştir.
- Kaydedilen operatör fonksiyonlarının ismi alet çantası tarafından değiştirilir, böylelikle evrimsel algoritmalar operatör isimlerine bağlı kalmaz.
- Değerlendirme(amaç) fonksiyonunu kullanıcının oluşturması gerekir.
- Uygunluk değerleri yinelenebilir olmalıdır, bu nedenle amaç fonksiyonundan tuple dönmelidir

Operatör örneği

```
def evaluate(individual): # Amaç fonksiyonu
        return sum(individual), # Geriye tuple döner

toolbox.register("mate", tools.cxTwoPoint)
toolbox.register("mutate", tools.mutGaussian, mu=0, sigma=1, indpb=0.1)
toolbox.register("select", tools.selTournament, tournsize=3)
toolbox.register("evaluate", evaluate)
```

Operatörler

Çaprazlama Operatörleri

- `cxOnePoint`: Tek noktalı çaprazlama(bit, tamsayı, gerçel sayı)
- `cxTwoPoint`: İki noktalı çaprazlama(bit, tamsayı, gerçel sayı)
- `cxUniform`: Tekörnek çaprazlama(bit, tamsayı, gerçel sayı)
- `cxPartialyMatched`: Kısmi eşleşmiş çaprazlama(permütasyon temsili)[8]
- `cxUniformPartialyMatched`: Tekörnek PMX(permütasyon)[3]
- `cxOrdered`: Sıralı çaprazlama(permütasyon)[7]
- `cxBlend`: Karışım çaprazlama(gerçel sayı)
- `cxESBlend` Evrim stratejileri için karışım çaprazlama(gerçel sayı)
- `cxESTwoPoint`: Evrim stratejileri için iki noktalı çaprazlama(gerçel sayı)
- `cxSimulatedBinary`: Gerçel sayılar
- `cxSimulatedBinaryBounded`: Gerçel sayılar
- `cxMessyOnePoint`: Tek noktalı, birey boyutu değişir

Operatörler

Mutasyon Operatörleri

- `mutGaussian`: μ ortalama, σ standart sapmaya sahip Gauss mutasyonu(gerçel sayılar)
- `mutShuffleIndexes`: İndisleri karıştırır(genellikle permütasyon veya indis vektörü)
- `mutFlipBit`: Bitleri tersine çevirir(bit veya boolean)
- `mutUniformInt`: Alt ve üst aralık arasında rastgele tamsayı
- `mutPolynomialBounded`: orjinal NSGA-II yöntemindeki mutasyon[5]
- `mutESLogNormal`: Evrim stratejileri[2]

Algoritmalar

Varyasyonlar

- Varyasyonlar, daha karmaşık algoritmalar oluşturmak için kullanılabilecek algoritma parçalarıdır
- `deap.algorithms.varAnd`: Çaprazlama **ve** mutasyon uygulanır
  - Parametreleri
    - `population`: Varyasyona tabi tutulacak birey listesi
    - `toolbox`: EA operatörlerini içeren alet kutusu
    - `cxpb`: Çaprazlama olasılığı
    - `mutpb`: Mutasyon olasılığı
  - Geri dönüş değeri: Ebeveynlerden bağımsız varyasyona uğramış bireyler

Algoritmalar

Varyasyonlar

- `deap.algorithms.varOr`: Çaprazlama **veya** mutasyon uygulanır
  - Parametreleri
    - `population`: Varyasyona tabi tutulacak birey listesi
    - `toolbox`: EA operatörlerini içeren alet kutusu
    - `lambda_`: Üretilecek çocuk sayısı
    - `cxpb`: Çaprazlama olasılığı
    - `mutpb`: Mutasyon olasılığı
  - Geri dönüş değeri: Ebeveynlerden bağımsız varyasyona uğramış bireyler

Algoritmalar

deap.algorithms.eaSimple

- En basit evrimsel algoritma türünü gerçekleştirir[1]
- Sözde kodu aşağıda verilmiştir

eaSimple Sözde Kodu

```
evaluate(population)
for g in range(ngen):
        population = select(population, len(population))
        offspring = varAnd(population, toolbox, cxpb, mutpb)
        evaluate(offspring)
        population = offspring
```

Algoritmalar

`deap.algorithms.eaSimple`

- Parametreleri:
  - `population`: Birey listesi
  - `toolbox`: EA operatörlerini içeren alet kutusu
  - `cxpb`: Çaprazlama olasılığı
  - `mutpb`: Mutasyon olasılığı
  - `ngen`: Nesil sayısı
  - `stats`: `Statistics` nesnesi, eğitim süresince güncellenir
  - `halloffame`: Tüm eğitim boyunca en iyi bireyleri saklayan `HallOfFame` nesnesi
  - `verbose`: İstatistiklerin gösterilip gösterilmeyeceği
- Geri dönüş değerleri:
  - Son popülasyon
  - Evrimsel hesaplamanın istatistiklerini tutan `deap.tools.Logbook` nesnesi

Algoritmalar

`deap.algorithms.eaMuPlusLambda`

- $(\mu + \lambda)$ evrimsel algoritmasını gerçekleştirir
- Sözde kodu aşağıda verilmiştir

eaMuPlusLambda Sözde Kodu

```
evaluate(population)
for g in range(ngen):
        offspring = varOr(population, toolbox, lambda_, cxpb,
        ↪  mutpb)
        evaluate(offspring)
        population = select(population + offspring, mu)
```

Algoritmalar

`deap.algorithms.eaSimple`

- Parametreleri:
  - `population`: Birey listesi
  - `toolbox`: EA operatörlerini içeren alet kutusu
  - `mu`: Sonraki nesle seçilecek birey sayısı
  - `lambda_`: Her nesilde üretilecek çocuk sayısı
  - `cxpb`: Çaprazlama olasılığı
  - `mutpb`: Mutasyon olasılığı
  - `ngen`: Nesil sayısı
  - `stats`: `Statistics` nesnesi, eğitim süresince güncellenir
  - `halloffame`: Tüm eğitim boyunca en iyi bireyleri saklayan `HallOfFame` nesnesi
  - `verbose`: İstatistiklerin gösterilip gösterilmeyeceği
- Geri dönüş değerleri:
  - Son popülasyon
  - Evrimsel hesaplamanın istatistiklerini tutan `deap.tools.Logbook` nesnesi

Algoritmalar

`deap.algorithms.eaMuCommaLambda`

- (μ, λ) evrimsel algoritmasını gerçekleştirir
- Sözde kodu aşağıda verilmiştir

eaMuCommaLambda Sözde Kodu

```
evaluate(population)
for g in range(ngen):
        offspring = varOr(population, toolbox, lambda_, cxpb,
        ↪  mutpb)
        evaluate(offspring)
        population = select(offspring, mu)
```

- Parametreleri $(\mu + \lambda)$ algoritması ile aynıdır

Algoritmalar

deap.algorithms.eaGenerateUpdate

- Bu algoritma, [4]'de önerilen sor-söyle modelini uygular. Sor, oluşturma(generate); söyle ise güncelleme(update) olarak adlandırılır.
- Alet kutusunda `generate` ve `update` içinde tanımlı olmalıdır.

eaGenerateUpdate Sözde Kodu

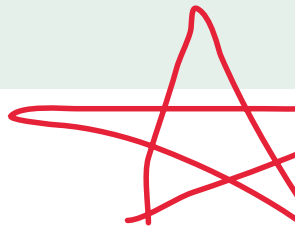
```
for g in range(ngen):
        population = toolbox.generate()
        evaluate(population)
        toolbox.update(population)
```

Algoritmalar

`deap.algorithms.eaGenerateUpdate`

- Parametreleri:
  - `toolbox`: Alet kutusu
  - `ngen`: Nesil sayısı
  - `stats`: İstatistik nesnesi
  - `halloffame`: En iyi bireylerin saklandığı liste
  - `verbose`: İstatistiklerin gösterilip gösterilmeyeceği
- Geri dönüş değerleri
  - Son popülasyon
  - Evrimsel hesaplamanın istatistiklerini tutan `deap.tools.Logbook` nesnesi

Çözüm Adımları



- DEAP kütüphanesi ile bir problemi çözmek için aşağıdaki adımlar takip edilmelidir
  - Kullanılacak paketlerin içe aktarılması(import)
  - Uygunluk ve birey türlerinin oluşturulması(`creator.create`)
  - Alet kutusu oluşturma(`toolbox`)
  - Birey ve popülasyon fonksiyonlarının alet kutusuna kaydedilmesi(`toolbox.register`)
  - Değerlendirme(Amaç) fonksiyonunun yazılması(`evaluate`)
  - Değerlendirme, çaprazlama, mutasyon ve seçme işlemlerinin alet kutusuna kaydedilmesi(`toolbox.register`)
  - Popülasyonun oluşturulup seçilen algoritma ile eğitilmesi(`algorithms`)

One Max Problemi

Problem Tanımı

- Basit ve evrimsel hesaplama topluluğunda sıklıkla kullanılan bir problemdir
- Rastgele 0 ve 1'lerden oluşan bir tamsayı vektörü olsun
- Popülasyonun tamamen 1'lerden oluşacak şekilde evrilmesini sağlanır(Hiç 0 yok)

One Max Problemi

İçe Aktarma ve Tür Oluşturma

İçe akrarma

```
import random # Rastgele değer üretimi için
from deap import base # Fitness ve Toolbox sınıfları için
from deap import creator # Tür oluşturmak için
from deap import tools # Çeşitli operatörler
```

Türleri oluşturma

```
creator.create("FitnessMax", base.Fitness, weights=(1.0,))
creator.create("Individual", list, fitness=creator.FitnessMax)
```

One Max Problemi

Alet kutusu ve kaydetme

```
toolbox = base.Toolbox()
# Özellik üreteci
toolbox.register("attr_bool", random.randint, 0, 1)
# Yapı başlatıcıları
toolbox.register("individual", tools.initRepeat,
↪  creator.Individual, toolbox.attr_bool, 100)
toolbox.register("population", tools.initRepeat, list,
↪  toolbox.individual)
```

```
>>> toolbox.attr_bool()
0
>>> toolbox.attr_bool()
1
>>> toolbox.individual()
[1, 1, 0, 0, 1, 1, 1, 0, 0, 1, 1, 1, 0, 1, 0, 0, 1, 0, 1, 1, 0, 1, 1, 0, 0, 0,
↪  0, 0, 1, 0, 1, 1, 1, 0, 0, 0, 0, 0, 1, 1, 0, 1, 1, 0, 0, 0, 1, 0, 1, 1, 1,
↪  1, 0, 0, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 1, 1, 0, 0, 1, 1, 1, 0,
↪  1, 1, 1, 1, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 0, 1, 0, 1]
```

One Max Problemi

Değerlendirme Fonksiyonu ve Genetik Operatörler

Değerlendirme fonksiyonu

```
def evalOneMax(individual):
        return sum(individual), # Tuple olmalı
```

Genetik operatörler

```
toolbox.register("evaluate", evalOneMax)
toolbox.register("mate", tools.cxTwoPoint)
toolbox.register("mutate", tools.mutFlipBit, indpb=0.05)
toolbox.register("select", tools.selTournament, tournsize=3)
```

One Max Problemi

Popülasyonun Oluşturulması ve Değerlendirilmesi

Popülasyonun Oluşturulması ve Değerlendirilmesi

```
pop = toolbox.population(n=300)
# Tüm popülasyonu değerlendir
fitnesses = list(map(toolbox.evaluate, pop))
for ind, fit in zip(pop, fitnesses):
        ind.fitness.values = fit
```

Olasılıkların belirlenmesi ve uygunluk değerlerinin alınması

```
# Çaprazlama ve mutasyon olasılıklarının belirlenmesi
CXPB, MUTPB = 0.5, 0.2
# Uygunluk değerlerinin çıkartılması
fits = [ind.fitness.values[0] for ind in pop]
```

One Max Problemi

EA Adımları

EA Adımları

```
g = 0 # Nesil sayacı
while max(fits) < 100 and g < 1000:
        g = g + 1 # Nesil sayısını artır
        print("-- Generation %i --" % g)
        # Sonraki nesildeki bireyleri seç
        offspring = toolbox.select(pop, len(pop))
        # Seçilen bireyleri klonla
        offspring = list(map(toolbox.clone, offspring))
        # Çaprazlama ve mutasyon uygula
        for child1, child2 in zip(offspring[::2], offspring[1::2]):
                if random.random() < CXPB:
                        toolbox.mate(child1, child2)
                        del child1.fitness.values
                        del child2.fitness.values
        for mutant in offspring:
                if random.random() < MUTPB:
                        toolbox.mutate(mutant)
                        del mutant.fitness.values
```

One Max Problemi

EA Adımları

EA Adımları

```
# Geçersiz uygunluk değeri olanları yeniden değerlendir
invalid_ind = [ind for ind in offspring if not ind.fitness.valid]
fitnesses = map(toolbox.evaluate, invalid_ind)
for ind, fit in zip(invalid_ind, fitnesses):
        ind.fitness.values = fit
pop[:] = offspring # Çocuklar yeni nesli oluşturur

# Uygunluk değerlerini listeye al ve istatistikleri yazdır
fits = [ind.fitness.values[0] for ind in pop]
length = len(pop)
mean = sum(fits) / length
sum2 = sum(x*x for x in fits)
std = abs(sum2 / length - mean**2)**0.5
print("  Min %s" % min(fits))
print("  Max %s" % max(fits))
print("  Avg %s" % mean)
print("  Std %s" % std)
```

One Max Problemi

Kısa Sürüm

One Max Kısa Sürüm

```
pop = toolbox.population(n=300)
hof = tools.HallOfFame(1)
stats = tools.Statistics(lambda ind: ind.fitness.values)
stats.register("avg", numpy.mean)
stats.register("std", numpy.std)
stats.register("min", numpy.min)
stats.register("max", numpy.max)

pop, log = algorithms.eaSimple(pop, toolbox, cxpb=0.5, mutpb=0.2,
↪  ngen=40, stats=stats, halloffame=hof, verbose=True)
```

0-1 Sırt Çantası Problemi

Problem Dosyaları

https://people.sc.fsu.edu/~jburkardt/datasets/knapsack_01/knapsack_01.html

p01_c.txt

165

p01_p.txt

92
57
49
68
60
43
67
84
87
72

p01_w.txt

23
31
29
44
53
38
63
85
89
82

p01_s.txt

1
1
1
1
0
1
0
0
0
0

0-1 Sırt Çantası Problemi

Tür Tanımları ve Alet Kutusu

Tür Tanımları ve Alet Kutusu

```
creator.create("Fitness", base.Fitness, weights=(1.0,))
creator.create("Individual", list, fitness=creator.Fitness)
toolbox = base.Toolbox()
toolbox.register("attr_rand", random.randint, 0, 1)
toolbox.register("individual", tools.initRepeat,
↪  creator.Individual, toolbox.attr_rand, len(weights))
toolbox.register("population", tools.initRepeat, list,
↪  toolbox.individual)
toolbox.register("evaluate", evaluate, capacity, prices, weights)
↪  # ekstra parametreler
```

0-1 Sırt Çantası Problemi

Değerlendirme Fonksiyonu

Değerlendirme Fonksiyonu

```
def evaluate(capacity, prices, weights, candidate):
        total_weight = 0
        total_price = 0
        for i in range(len(candidate)):
                total_weight += candidate[i] * weights[i]
                total_price += candidate[i] * prices[i]
        if total_weight > capacity:
                return 0,
        return total_price,
```

0-1 Sırt Çantası Problemi

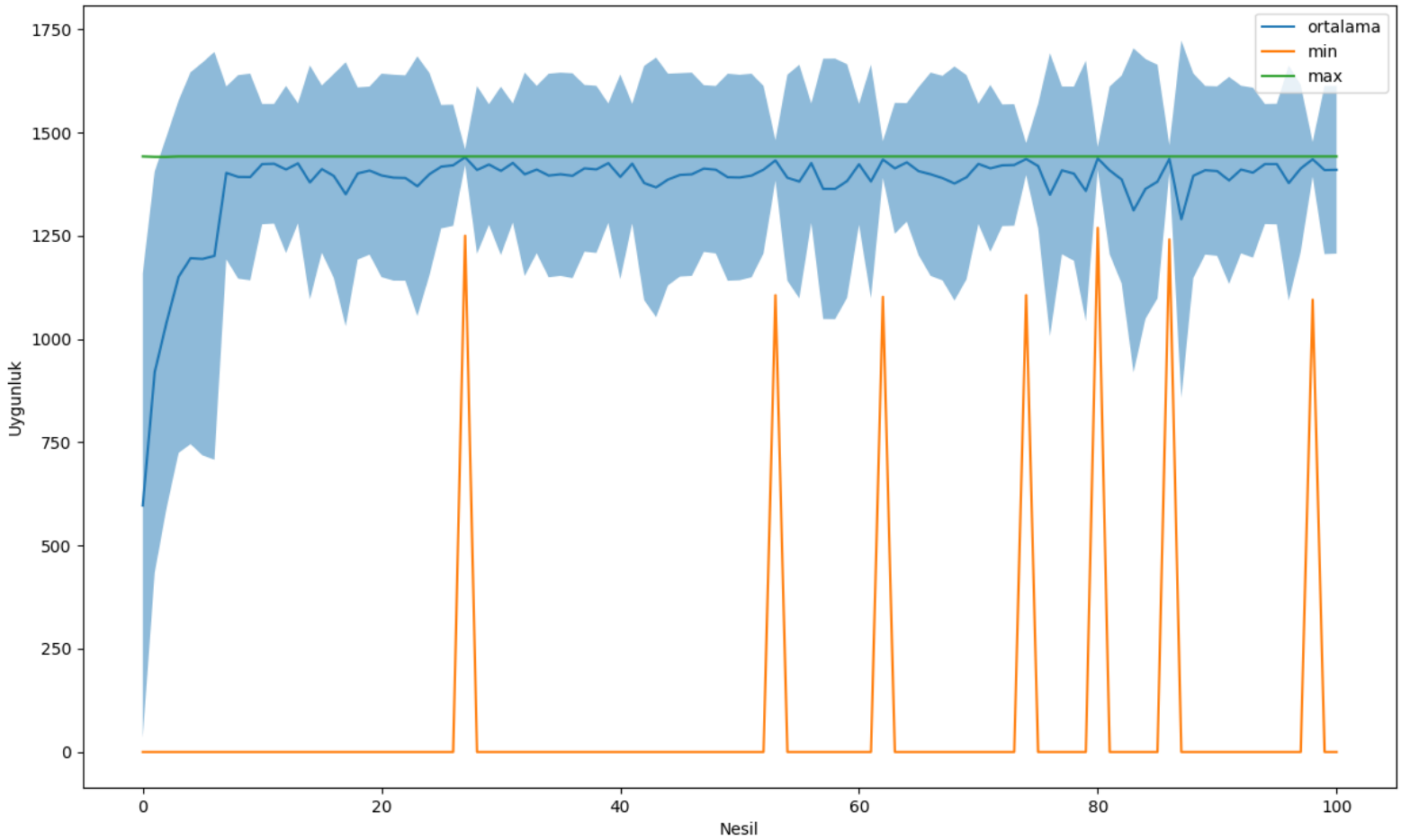
Eğitim Grafiği Çizdirme(matplotlib)

Grafik çizdirme

```
def plot_statistics(logbook):
        gen, min_, max_, avg, std = logbook.select("gen", "min",
        ↪  "max", "avg", "std")
        fig, ax = plt.subplots()
        ax.set_xlabel("Nesil")
        ax.set_ylabel("Uygunluk")
        ax.plot(gen, avg, label="ortalama")
        avg = np.array(avg) # Numpy dizisine dönüştür
        std = np.array(std) # Numpy dizisine dönüştür
        ax.fill_between(gen,avg-std,avg+std,alpha=.5)
        ax.plot(gen, min_, label="min")
        ax.plot(gen, max_, label="max")
        ax.legend()
        plt.show()
```

0-1 Sırt Çantası Problemi

Eğitim Grafiği(Problem 7)



Gezgin Satıcı Problemi(Türkiye)

Dosya Okuma ve Tür Tanımlama

- 81 il için TSP
- İller arası mesafe bir csv dosyasında kayıtlı(matris olarak)

Dosya Okuma ve Tür Tanımlama

```
mesafe=list()
f=open('ilmesafe2.csv')
for satir in f.read().split('\n'):
mesafe.append(list(map(lambda x:0 if x=='' else
↪  int(x),satir.split(';'))))
IND_SIZE=len(mesafe) # Birey boyutu
creator.create("FitnessMin", base.Fitness, weights=(-1.0,))
creator.create("Individual", list, fitness=creator.FitnessMin)
```

Gezgin Satıcı Problemi

Alet Kutusu ve Değerlendirme Fonksiyonu

Dosya Okuma ve Tür Tanımlama

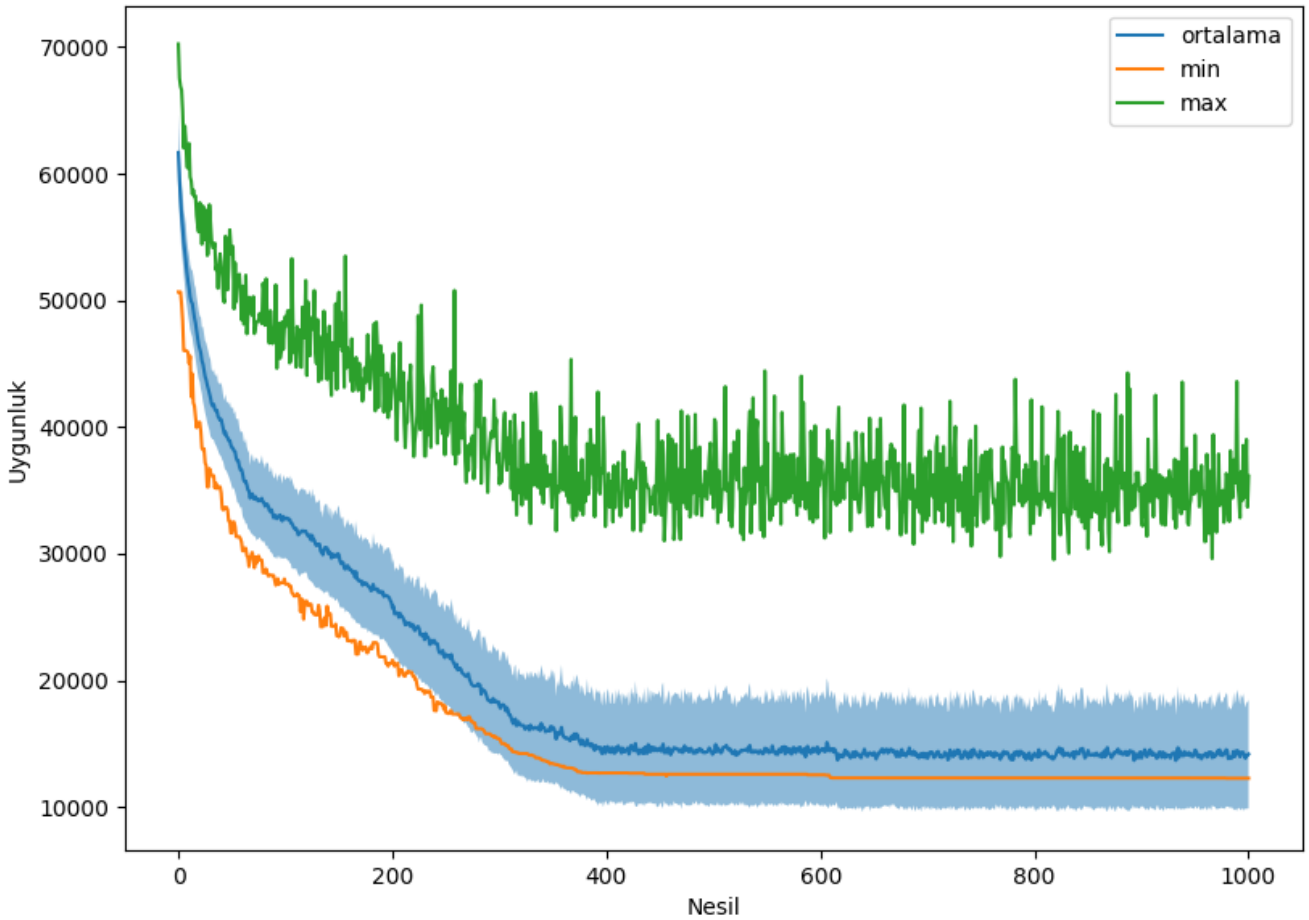
```
toolbox.register("indices", random.sample, range(IND_SIZE), IND_SIZE)
toolbox.register("individual", tools.initIterate, creator.Individual,
↪  toolbox.indices)
toolbox.register("population", tools.initRepeat, list, toolbox.individual)

def evalTSP(ind):
        l=0
        for i in range(len(ind)-1):
                l += mesafe[ind[i]][ind[i+1]]
        l += mesafe[ind[0]][ind[-1]]
        return l,

toolbox.register("mate", tools.cxOrdered)
toolbox.register("mutate", tools.mutShuffleIndexes, indpb=0.05)
toolbox.register("select", tools.selTournament, tournsize=3)
toolbox.register("evaluate", evalTSP)
```

Gezgin Satıcı Problemi

Eğitim Grafiği



Gezgin Satıcı Problemi

Eğitim Sonuçları

- Nesil sayısı: 1000
- Popülasyon boyutu: 500
- cxpb: 0.7
- mutpb: 0.2
- En iyi uygunluk: **12281**

Çoklu İşlem

Çoklu işlem için gerekenler

```
from multiprocessing import Pool
pool = Pool() # İşçi havuzunu oluştur
toolbox.register("map", pool.map)
```

Kaynaklar I

[1] Thomas Bäck, David B Fogel, and Zbigniew Michalewicz.
Evolutionary computation 1: Basic algorithms and operators.
CRC press, 2018.

[2] Hans-Georg Beyer and Hans-Paul Schwefel.
Evolution strategies–a comprehensive introduction.
Natural computing, 1(1):3–52, 2002.

[3] Vincent A Cicirello and Stephen F Smith.
Modeling ga performance for control parameter optimization.
In *GECCO-2000: Proceedings of the Genetic and Evolutionary Computation Conference*, pages 235–242, 2000.

Kaynaklar II

[4] Yann Collette, Nikolaus Hansen, Gilles Pujol, Daniel Salazar Aponte, and Rodolphe Le Riche.
Object-oriented programming of optimizers–examples in scilab.
Multidisciplinary Design Optimization in Computational Mechanics, pages 499–538, 2013.

[5] Kalyanmoy Deb, Samir Agrawal, Amrit Pratap, and Tanaka Meyarivan.

A fast elitist non-dominated sorting genetic algorithm for multi-objective optimization: Nsga-ii.
In *International conference on parallel problem solving from nature*, pages 849–858. Springer, 2000.

Kaynaklar III

[6] Félix-Antoine Fortin, François-Michel De Rainville, Marc-André Gardner, Marc Parizeau, and Christian Gagné.
DEAP: Evolutionary algorithms made easy.
Journal of Machine Learning Research, 13:2171–2175, jul 2012.

[7] D. E. Goldberg.
Genetic algorithms in search, optimization and machine learning.
Addison-Wesley, 1989.

[8] David E Goldberg, Robert Lingle, et al.
Alleles, loci, and the traveling salesman problem.
In *Proceedings of an international conference on genetic algorithms and their applications*, volume 154, pages 154–159. Carnegie-Mellon University Pittsburgh, PA, 1985.

Kaynaklar IV

[9] Yannick Hold-Geoffroy, Olivier Gagnon, and Marc Parizeau.
Once you scoop, no need to fork.
In *Proceedings of the 2014 Annual Conference on Extreme Science and Engineering Discovery Environment*, page 60. ACM, 2014.