Практическое занятие. Генетический алгоритм поэтапно.

Генетический алгоритм реализуется поэтапно. Этапы таковы.

1. Сгенерировать начальную популяцию.
2. Выбрать из популяции лучших представителей.
3. Породить из лучших представителей потомков.
4. Если улучшить целевую функцию удалось, то скорректировать популяцию и продолжить с шага 1, в противном случае закончить (мы здесь ограничимся этими шагами).

Нам нужно для работы инсталлировать пакет DEAP. Это делаем так

```
pip install deap
```

Либо в Анаконде

```
conda install -c conda-forge deap
```

Подключите импорты

```
import random

from deap import base, creator, tools
```

Определите выражение для целевой функции

```
def eval_func(individual):

   target_sum = 15

   return len(individual) - abs(sum(individual) - target_sum),
```

Что, по-вашему она делает?

Создадим управляющую панель

```
def create_toolbox(num_bits):

   creator.create("FitnessMax", base.Fitness, weights=(1.0,))

   creator.create("Individual", list, fitness=creator.FitnessMax)
```

Инициализируем панель

```
    toolbox = base.Toolbox()
toolbox.register("attr_bool", random.randint, 0, 1)
toolbox.register("individual", tools.initRepeat, creator.Individual,
    toolbox.attr_bool, num_bits)
toolbox.register("population", tools.initRepeat, list, toolbox.individual)
```

Назначим операции оценки популяции функцию – eval_func

```
toolbox.register("evaluate", eval_func)
```

То же самое для операции обмена генами –

```
toolbox.register("mate", tools.cxTwoPoint)
```

и для операции мутации –

```
toolbox.register("mutate", tools.mutFlipBit, indpb = 0.05)
```

Определим операцию для генерации потомков –

```
toolbox.register("select", tools.selTournament, tournsize = 3)
return toolbox
if __name__ == "__main__":
    num_bits = 45
    toolbox = create_toolbox(num_bits)
    random.seed(7)
    population = toolbox.population(n = 500)
    probab_crossing, probab_mutating = 0.5, 0.2
    num_generations = 10
    print('\nEvolution process starts')
```

Наконец, оценка всей популяции –

```
fitnesses = list(map(toolbox.evaluate, population))
for ind, fit in zip(population, fitnesses):
    ind.fitness.values = fit
print('\nEvaluated', len(population), 'individuals')
```

Вывод популяций –

```
for g in range(num_generations):
    print("\n- Generation", g)
```

Важная операция – выбор новой популяции –

```
offspring = toolbox.select(population, len(population))
```

Получение потомков-клонов –

```
offspring = list(map(toolbox.clone, offspring))
```

Обмен генами и мутация –

```
for child1, child2 in zip(offspring[::2], offspring[1::2]):
   if random.random() < probab_crossing:
   toolbox.mate(child1, child2)
```

Удаление оценки потомков

```
del child1.fitness.values
del child2.fitness.values
```

Применение мутации –

```
for mutant in offspring:
   if random.random() < probab_mutating:
   toolbox.mutate(mutant)
   del mutant.fitness.values
```

Оценка индивидуумов с плохой генетикой –

```
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
print('Evaluated', len(invalid_ind), 'individuals')
```

Обновление популяции –

```
population[:] = offspring
```

Вывод статистики для текущей популяции –

```
fits = [ind.fitness.values[0] for ind in population]
length = len(population)
mean = sum(fits) / length
sum2 = sum(x*x for x in fits)
```

```
std = abs(sum2 / length - mean**2)**0.5

print('Min =', min(fits), ', Max =', max(fits))

print('Average =', round(mean, 2), ', Standard deviation =',

round(std, 2))

print("\n- Evolution ends")
```

Вывод окончательного результата –

```
   best_ind = tools.selBest(population, 1)[0]
   print('\nBest individual:\n', best_ind)
   print('\nNumber of ones:', sum(best_ind))
Following would be the output:
Evolution process starts
Evaluated 500 individuals
- Generation 0
Evaluated 295 individuals
Min = 32.0 , Max = 45.0
Average = 40.29 , Standard deviation = 2.61
- Generation 1
Evaluated 292 individuals
Min = 34.0 , Max = 45.0
Average = 42.35 , Standard deviation = 1.91
- Generation 2
Evaluated 277 individuals
Min = 37.0 , Max = 45.0
Average = 43.39 , Standard deviation = 1.46
… … … …
- Generation 9
Evaluated 299 individuals
Min = 40.0 , Max = 45.0
Average = 44.12 , Standard deviation = 1.11
- Evolution ends
Best individual:
[0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 1,
 1, 0, 0, 0, 1, 0, 0, 0, 1, 1, 0, 0, 0, 1, 0, 0, 0,
 1, 0, 0, 1, 1, 1, 0, 0, 1, 0, 1]
Number of ones: 15
```

ЗАДАНИЕ.

Попробуйте все соединить воедино, чтобы получить работающий генетический алгоритм. В нашем примере он должен породить строку из нулей и единиц, содержащую 15 единиц (проверьте, правильно ли вы поняли целевую функцию).

Теперь измените целевую функцию таким образом, что 15 единиц прижимались к левой части строки, а нулевые элементы – к правой.