Algorytmy ewolucyjne

Zadanie 1

Sprawozdanie

Jakub Łaba

Politechnika Warszawska
Wydział Elektryczny
Listopad 2023

Spis treści

1 Polecenie

Napisać program umożliwiający znalezienie maksimum funkcji dopasowania jednej zmiennej określonej dla liczb całkowitych w zadanym zakresie przy pomocy elementarnego algorytmu genetycznego (redprodukcja z użyciem ruletki, krzyżowanie proste, mutacja równomierna). Program powinien umożliwiać użycie różnych funkcji dopasowania, populacji o różnej liczebności oraz różnych parametrów operacji genetycznych (krzyżowania i mutacji).
Program powinien zapewnić wizualizację wyników w postaci wykresów średniego, maksymalnego oraz minimalnego przystosowania dla kolejnych populacji oraz wykresu funkcji w zadanym przedziale.
Program przetestować dla funkcji:
$f(x) = -0.1x^2 + 4x + 7$ dla $x \in [-1, 41] \cap \mathbb{Z}$

2 Technologia

Jako język implementacji postanowiłem wybrać Python, ze względu na prostotę oraz szeroki wybór pakietów.

- DEAP (Distributed Evolutionary Algorithms in Python) – biblioteka zawierająca narzędzia do algorytmów ewolucyjnych
- numpy – każdemu znana biblioteka do obliczeń numerycznych, tutaj używana jedynie do liczenia statystyk do wykresu
- matplotlib – biblioteka do rysowania wykresów

3 Opis rozwiązania

3.1 Konfiguracja programu

Napisałem program w sposób pozwalający na dostosowanie podstawowych parametrów algorytmu: liczba generacji, rozmiar populacji, prawdopodobieństwa krzyżowania oraz mutacji. Można je dostosować, edytując plik `config.json`:

```
{
  "num_generations": 50,
  "population_size": 100,
  "crossover_rate": 0.7,
  "mutation_rate": 0.2
}
```

3.2 Definicja algorytmu ewolucyjnego w bibliotece DEAP

Biblioteka DEAP pozwala w bardzo prosty i szybki sposób prototypować algorytmy ewolucyjne – należy zdefiniować typ problemu, osobnika, oraz funkcję przystosowania.
Typ problemu (w tym przypadku poszukiwanie maksimum funkcji) oraz osobnika można zdefniować za pomocą funkcjonalności `creator`:

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

Następnie definiuję opakowanie (`evaluate`) na swoją funkcję przystosowania (`fitness_func`), aby była kompatybilna z kontraktem wymaganym przez bibliotekę:

```
def fitness_func(x: int) -> float:
    return -.1 * x ** 2 + 4 * x + 7
```

```
def evaluate(individual: List[int]) -> Tuple[float]:
    x = individual[0]
    return fitness_func(x),
```

Można zauważyć, że osobnik jest zdefiniowany w dość nieintuicyjny sposób – skoro poszukiwane są pojedynczych wartości, dlaczego osobnik reprezentowany jest w formie listy? Jest to niestety ograniczenie wynikające z biblioteki, które obchodzę używając małych list (2 elementy - minimalny dopuszczalny przez bibliotekę rozmiar), a następnie ignorując drugi element z nich. Nieintuicyjny może być też typ zwracany przez funkcję `evaluate` – niestety również jest to wymaganie biblioteki, ta funkcja musi zwracać typ `Tuple`. Więcej na ten temat można poczytać w **dokumentacji**.

3.3 Konfiguracja właściwości algorytmu w bibliotece DEAP

Parametry algorytmu, takie jak populacja początkowa, metoda krzyżowania, itd. można dowolnie dostosować za pomocą funkcjonalności `toolbox`. Skrócony przykład, pokazujący ustawienia: metoda krzyżowania (proste), rodzaj mutacji (równomierna), metoda selekcji (ruletka).

```
toolbox = base.Toolbox()
toolbox.register("mate", tools.cxTwoPoint)
toolbox.register("mutate", tools.mutUniformInt, low=LOW, up=UP, indpb=mut_rate)
toolbox.register("select", tools.selRoulette)
```

3.4 Uruchomienie algorytmu

Moduł `algorithms` posiada wiele gotowych algorytmów, na potrzeby tego zadania można wykorzystać `eaSimple` z dostosowanymi parametrami, opisanymi w poprzednich podpunktach sprawozdania. Biblioteka jest na tyle wygodna, że algorytm od razu generuje obiekt `Logbook` zawierający dane o przebiegu algorytmu, który można potem wygodnie wykorzystać do narysowania wykresów.

```
_, logbook = algorithms.eaSimple(
    population, toolbox,
    cxpb=crossover_rate,
    mutpb=mutation_rate,
    ngen=num_generations,
    stats=stats,
    verbose=True
)
```