Politechnika Warszawska

Wydział Elektroniki i Technik Informacyjnych

Wstęp do Sztucznej Inteligencji

ćwiczenie 2 - Algorytmy ewolucyjne i genetyczne

Autor:

Jarosław Jaworski 342189

Data oddania ćwiczenia:

1.04.2025

SPIS TREŚCI

1 Cel ćwiczenia 2

2 Rozwiązanie trywialne 2

3 Górne ograniczenie wyniku 2

4 Wyniki pomiarów 3

4.1 Rozwiązanie losowe 3

5 Wnioski 4

1 Cel ćwiczenia

Celem ćwiczenia była analiza algorytmów genetycznych z rodziny algorytmów ewolucyjnych oraz zastosowanie podstawowych metod podczas realizacji przykładu. W ćwiczeniu należało rozwiązać zadanie maksymalizacji.

2 Rozwiązanie trywialne

Ćwiczenie rozpoczęto od oceny rozwiązania trywialnego. Ze względu na logikę zdobywania punktów, najprostszym sposobem na uzyskanie zadowalającego wyniku jest zaimplementowanie **szachownicy**. W tej formule, gdy naprzemiennie ustawiamy zera i jedynki, otrzymujemy dokładnie połowę pól spełniającą warunek uzyskania punktu. Oznacza to, że jeżeli posiadamy 400 pól, to połowa zapewnia punkty. Rozwiązanie to traktuje się jako odniesienie dla otrzymywanych algorytmem genetycznym. Zależność na liczbę punktów rozwiązania trywialnego opisuje wzór (1)

$$avg\_max\_points = vector\_size/2 \tag{1}$$

,zatem $avg\_max\_points = 200$.

3 Górne ograniczenie wyniku

Do znalezienia górnego maksymalnego ograniczenia wyniku potrzeba głębszej analizy istoty problemu. Porządanym efektem jest, aby każda jedynka miała w swoim otoczeniu jak największą liczbę zer. Niestety, ze względu na nieliczenie punktów "na ukos" nieodpowiednim rozwiązaniem jest stosowanie np. wypełnienia przez znak plus (z jedynką w środku i zerami w ramionach) ze względu na pozostawiane puste przestrzenie. Zagęszczenie znaku doprowadza do struktury

niemalże identycznej jak dla szachownicy. Kolejnym sposobem jest zaimplementowanie tego samego ciągu znaków w rzędzie dla całej macierzy. Minimalizacja jedynek w rzędzie przy maksymalizacji zdobytych punkt przez zera prowadzi do jednoznacznego rozwiązania maksymalnego. Wypełnienie dostępnej przestrzeni poprzez powielanie najlepszego ciągu znaków zagwarantowałoby uzyskanie maksymalnego możliwego wyniku. Implementacja wygląda następująco:

Rząd 1: 0 1 0 0 1 0 0 1 ...

Rząd 2: 0 1 0 0 1 0 0 1 ...

...

Wtedy, maksymalną liczbę punktów możemy wyznaczyć na podstawie wzoru (2)

$$max\_points = zeroes\_per\_row * rows\_num \tag{2}$$

Tym sposobem łatwo policzyć, że maksymalna liczba punktów do uzyskania na płaszczyźnie wynosi $max\_points = 260$.

4 Wyniki pomiarów

4.1 Rozwiązanie losowe

Rozwiązanie losowe polegało na inicjowaniu pierwotnej populacji poprzez funkcję losową i stosowanie na niej algorytmu genetycznego. Dzięki temu zamiast zdeterminowanych wyników otrzymywano realistyczną informację o efektywności algorytmu dla różnych "punktów startowych". Testy algorytmu dla domyślnych parametrów wykonano **dla 150 różnych populacji startowych wygenerowanych losowo**. Tabela 1. przedstawia wyniki pomiarów.

Tabela 1. Wyniki pomiarów dla rozwiązania losowego

Parametr	**Wartość**
Średnia	216,03
Mediana	216
Odchylenie std.	2,74

5 Wnioski

Z punktu widzenia działania algorytmu, jest to niezawodny sposób na otrzymywanie wyników **conajmniej takich jak dla rozwiązania trywialnego**. W wielu iteracjach algorytm potrafi znaleźć zadowalające rozwiązanie. Przy losowej populacji początkowej, wynik zmierza do tej samej wartości końcowej z niewielkimi odchyleniami. Oznacza to, że algorytm bez dodatkowej pomocy (np. w postaci hiperparametru) nie jest w stanie osiągnąć lepszego wyniku niż pewna wartość (np. tu: średnio wynik 216).

W porównaniu z rozwiązaniem najlepszym, algorytm nie jest w stanie samodzielnie go znaleźć. W przypadku ćwiczenia otrzymane wyniki nie były nawet blisko rozwiązania bliskiego optymalnemu. Jest to powiązane z wysoką złożonością problemu znajdowania optimum dla tablicy elementów. Problem charakteryzuje się złożonością wykładniczą dla standardowych metod znajdowania ekstremum, stąd wiązałoby się to z nierealizowalnie długimi czasami egzekuzji programu.

Algorytm genetyczny z rodziny ewolucyjnych stanowi dobre rozwiązanie dla znajdowania esktremum złożonej funkcji celu. Jego rozwiązanie jest conajmniej trywialnym, jednak nie działa on wystarczająco dobrze aby dotrzeć do

optymalnego rozwiązania.

Istnieją pewne ulepszenia rozwiązania, które zaimplementowane w kodzie prawdopodobnie zbliżyły wyniki do optimum (nieznacznie, lecz zbliżyłyby). Przykładowo - funkcja selekcji naturalnej została w ramach ćwiczenia zaimplementowana metodą ruletki. Natomiast, można tę metodę dostosować poprzez np. dodanie przeżywalności najlepszych osobników starszej generacji. Wymiana całej generacji naraz (tak jak przeprowadzono to w ćwiczeniu) może wiązać się ze zmierzaniem do tego samego znalezionego optimum lokalnego (bo z generacji na generację nie osobników "dominujących" decydujących o dalszym kierunku przejścia algorytmu).