Algorytm genetyczny rozwiązujący problem plecakowy

1 Cel

Celem zadania jest zaimplementowanie algorytmu rozwiązującego dyskretny problem plecakowy. Zalecanym językiem jest Python. Zakazane się stosowanie gotowych bibliotek. Program powinien być czytelny, napisany z podziałem na funkcje, najlepiej w sposób obiektowy.

2 Opis problemu

Problem plecakowy (ang. Knapsack problem) jest klasycznym problemem optymalizacyjnym z teorii algorytmów i informatyki. Polega on na tym, że mając zbiór przedmiotów, z których każdy posiada określoną wagę i wartość, musimy wybrać podzbiór tych przedmiotów, aby zmaksymalizować łączną wartość przedmiotów w plecaku, nie przekraczając przy tym jego maksymalnej pojemności (czyli dopuszczalnej wagi).

Problem plecakowy znajduje zastosowanie w wielu dziedzinach, takich jak logistyka, zarządzanie zasobami, inwestycje czy planowanie produkcji. Jest to problem NP-trudny, co oznacza, że nie istnieje algorytm, który potrafi go rozwiązać w czasie wielomianowym dla dowolnej liczby przedmiotów. Jednak w praktyce stosuje się różne algorytmy heurystyczne lub techniki dynamicznego programowania, aby znaleźć rozwiązania bliskie optymalnym.

3 Algorytm genetyczny

Algorytm genetyczny jest to rodzaj heurystyki przeszukującej przestrzeń alternatywnych rozwiązań problemu w celu wyszukania rozwiązań najlepszych. Sposób działania algorytmów genetycznych przypomina zjawisko ewolucji biologicznej. Obecnie zalicza się go do grupy algorytmów ewolucyjnych.

Środowisko, w którym istnieje pewna populacja osobników, jest zdefiniowane przez problem, który należy rozwiązać. Każdy z osobników ma przypisany pewien zbiór chromosomów składający się z genów. Jest to genotyp danego osobnika i stanowi on podstawę do utworzenia fenotypu. Fenotyp to zbiór cech podlegających ocenie funkcji przystosowania modelującej środowisko. Innymi słowy - genotyp opisuje proponowane rozwiązanie problemu, a funkcja przystosowania ocenia, jak dobre jest to rozwiązanie. Każdy z osobników prezentuje lepsze lub gorsze rozwiązanie danego problemu. Każdy z osobników może posiadać kilka chromosomów, jednak bardzo często w algorytmach genetycznych wybrany osobnik posiada jeden chromosom. W literaturze oba pojęcia: chromosom oraz osobnik bardzo często używane są wymiennie.

Algorytm genetyczny posiada następujące kroki:

1. Wybór populacji początkowej
2. Ocena przystosowania osobników w populacji
3. Wyprowadzenie najlepszego osobnika
4. Sprawdzenie warunków zatrzymania
5. Selekcja chromosomów
6. Zastosowanie operatorów genetycznych
7. Utworzenie nowej populacji

3.1 Kodowanie informacji

Każdy chromosom będzie prezentowała jedno możliwe rozwiązanie tj. pewien podzbiór przedmiotów, które mogą zostać włożone do plecaka. Długość chromosomu będzie równa liczbie wszystkich przedmiotów. Pojedynczy gen będzie definiował, czy dany przedmiot znajduje się w plecaku (wartość genu 1) czy też danego przedmiotu w tym plecaku nie ma (wartość genu 0). Przykładowy chromosom, dla zadania, w którym występuje 15 przedmiotów, prezentuje diagram poniżej.

nr przedmiotu	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
chromosom	1	1	0	0	0	1	0	1	0	0	0	0	0	1	0

Chromosom koduje informacje, o tym że w plecaku znajdują się przedmioty: 1, 2, 6, 8, 14.

3.2 Operatory genetyczne

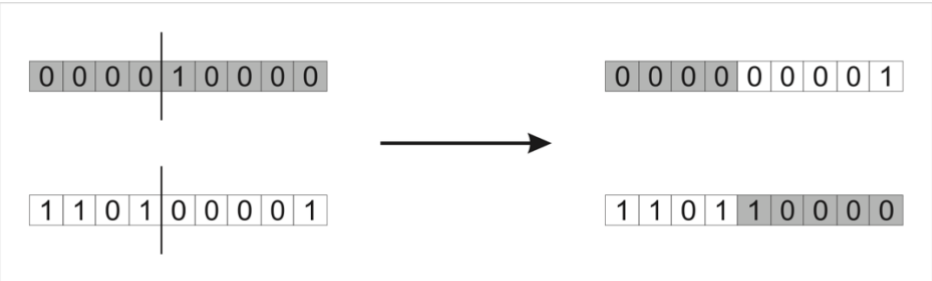
Operatory genetyczne mają na celu rekombinacje genów w chromosomach. Wyróżniamy dwa operatory genetyczne: krzyżowanie, mutację. Niekiedy podaje się jeszcze trzeci operator - inwersja. Najbardziej popularne i najczęściej stosowane są dwa pierwsze. Każdy z operatorów genetyczny wykonywany jest z pewnym prawdopodobieństwem, które definiujemy na początku programu i które jest stałe w trakcie jego trwania. Typowe zakresy prawdopodobieństw dla poszczególnych operatorów genetycznych są następujące:

- Krzyżowanie 0.5 – 1.0
- Mutacja 0.0 – 0.1

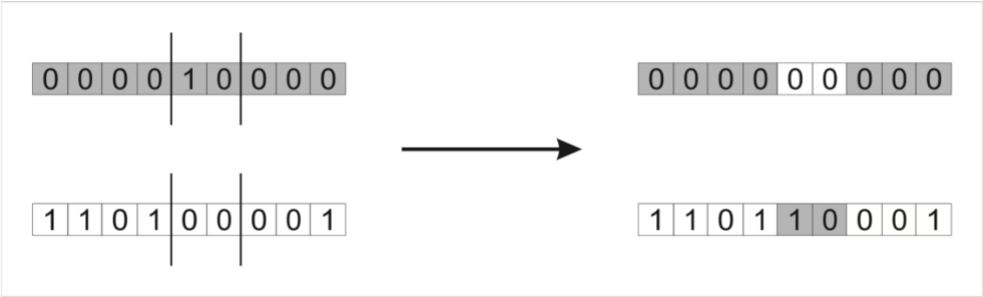
3.2.1 Krzyżowanie

Krzyżowanie jest operacją mającą na celu wymianę materiału genetycznego pomiędzy osobnikami.

Krzyżowanie jednopunktowe polega na wylosowaniu jednego punktu krzyżowania, a następnie wymianie materiału genetycznego w następujący sposób:



Krzyżowanie dwupunktowe jest wykonywane analogicznie do krzyżowania jednopunktowego z tą rocznicą, że losowane są dwa punkty:



3.2.2 Mutacja

Mutacja ma na celu wprowadzić różnorodność genetyczną populacji Prawdopodobieństwo mutacji wylicza się dla każdego genu. Mutacja w chromosomie binarnym polega na zmianie bitu na wartość przeciwną.

3.3 Funkcja przystosowania

Funkcja przystosowania ocenia jak dobrze dany osobnik jest przystosowany do środowiska (z punktu widzenia rozwiązywanego problemu). Dla dyskretnego problemu plecakowego będzie to wartość przedmiotów, jeżeli suma ich objętości nie przekracza maksymalnej objętości plecaka, w przeciwnym razie będzie to 0.

3.4 Funkcja selekcji

Metoda selekcji ma za zadanie wyselekcjonowanie najlepiej przystosowanych osobników przy jednoczesnym zachowaniu różnorodności genetycznej populacji. W wyniku selekcji dany osobnik może zostać wylosowany więcej niż jeden raz.

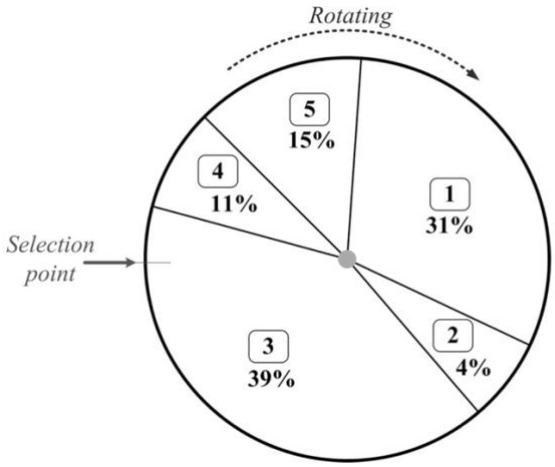
3.4.1 Selekcja koła ruletki

Polega na budowaniu wirtualnego koła, którego wycinki odpowiadają poszczególnym osobnikom. Im lepszy osobnik, tym większy wycinek koła zajmuje. Zajętość koła dla chromosomu ch_i w populacji złożonej z J osobników definiujemy z następującego wzoru:

$$p(ch_i) = \frac{f(ch_i)}{\sum_{j=1}^{J} f(ch_j)}$$

Rozmiar wycinków może zależeć od wartości funkcji oceny, jeśli wysoka wartość oceny oznacza wysokie przystosowanie. W takim układzie prawdopodobieństwo, że lepszy osobnik zostanie wybrany jako rodzic, jest większe. Niestety ewolucja przy takim algorytmie z każdym krokiem zwalnia. Jeżeli osobniki są podobne, to każdy dostaje równy

wycinek koła fortuny i presja selekcyjna spada. Algorytm słabiej rozróżnia osobniki dobre od słabszych.

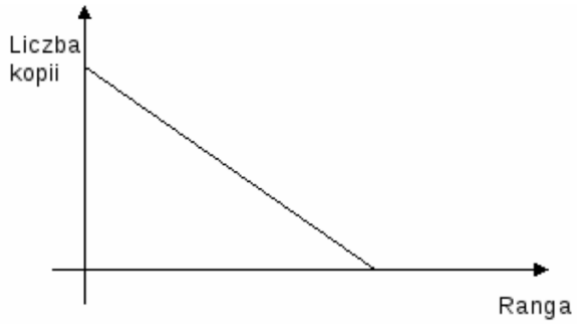


3.4.2 Selekcja turniejowa

Polega na wylosowaniu grupy k-elementowej z populacji a następnie wyborze osobnika o najlepszym przystosowaniu. Całą operację powtarzamy tyle razy, ile jest osobników w populacji.

3.4.3 Selekcja rankingowa

Osobniki ustawiane są kolejno zgodnie z wartością funkcji przystosowania - od najlepszego do naj- gorszego. Każdy osobnik ma numer określający jego pozycję na liście, czyli swoją rangę. Liczba kopii każdego osobnika wprowadzana do populacji jest zdefiniowana przez wcześniej ustaloną funkcję, która zależy od rangi osobnika.



4 Zadanie do wykonania

Podstawą oceny jest przesłanie sprawozdania wraz z plikami projektu.

4.1 Część programistyczna

Ocena maksymalna 3.5:

- Napisać funkcję importującą dane do problemu plecakowego z pliku tekstowego
- Napisać ogólny skrypt realizujący ewolucję algorytmu genetycznego. Parametrami dla skryptu powinny być m.in.: rozmiar populacji, liczba iteracji, funkcja przystosowania, funkcja selekcji, funkcje krzyżowania, funkcja mutacji.

- Napisać funkcję przystosowania odpowiednią dla problemu plecakowego.
- Napisać funkcję realizującą jednopunktowy operator krzyżowania.
- Napisać funkcję mutacji.
- Napisać funkcję realizującą selekcję ruletkową.

Ocena maksymalna 4.5 (dodatkowo):
- Napisać funkcję realizującą selekcję rankingową
- Napisać funkcję realizującą krzyżowanie dwupunktowe

Ocena maksymalna 5.0(dodatkowo):
- Napisać funkcję realizującą selekcję turniejową

4.2 Część eksperymentalna

Zaimportować zestaw danych. Dla każdego zbioru przeprowadzić ewolucję algorytmu. Zebrane wyniki przedstawić na wykresach (dla każdego zbioru osobny wykres). Wykresy powinny przedstawiać wartość funkcji dopasowania w zależności od iteracji algorytmu (po każdej iteracji odnotowujemy wartość funkcji dla najlepszego osobnika).

Ocena maksymalna 3.5:
- Wykresy dla różnych współczynników mutacji i krzyżowania (zaleca się jeden wykres dla jednego zbioru z kilkoma seriami danych).

Ocena maksymalna 4.5 (dodatkowo):
- Wykresy porównujące selekcję rankingową oraz ruletkową.
- Wykresy porównujące krzyżowanie jedno i dwupunktowe.

Ocena maksymalna 5.0 (dodatkowo):
- Wykresy porównujące selekcję rankingową, ruletkową oraz turniejową.

5 Zbiór danych

Do eksperymentu należy użyć danych zamieszczony na Moodle. Należy wybrać **4 zbiory** *low-dimensional* i **2 zbiory** *large_scale*. Wszystkie wyniki należy porównać z podanym optimum. Struktura danych:

```
Wielkość_problemu        pojmność
wartość_przedmiotu_1     waga_przedmiotu_1
...                      ...
wartość_przedmiotu_n     waga_przedmiotu_n
```

6 Źródło

http://wikizmsi.zut.edu.pl/uploads/1/15/Lab5.pdf