Przeszukiwanie i Optymalizacja

Projekt Wstępny - 21Z

Realizacja w zespole: Bratosiewicz Konrad, Marczuk Jakub
Temat projektu: SK.POP.2 - Złodziej

Treść zadania

Złodziej ukradł X gramów złota ze skarbca i wraca do domu pociągiem. Żeby uniknąć schwytania przez policję, musi zamienić złoto na banknoty, więc postanawia sprzedać złoto pasażerom pociągu. Zainteresowanych kupnem jest N pasażerów, każdy z nich zgadza się kupić Ai, gdzie i $\in$ (1, 2, …, N) gramów złota za Vi, i $\in$ (1, 2, …, N). Złodziej chce uciec przed policją, jednocześnie maksymalizując zysk.

Zaimplementuj program bazujący na algorytmie ewolucyjnym, który wskaże pasażerów, którym złodziej powinien sprzedać złoto oraz sumę wartości banknotów, którą zarobi. Zastosowanie i porównanie z innym algorytmem jest wskazane i będzie dodatkowym atutem przy ocenie projektu.

Analiza zadania

Mamy do czynienia z wariacją na temat problemu plecakowego. Z tego powodu oprócz algorytmu ewolucyjnego zamierzamy zaimplementować algorytm plecakowy oparty na programowaniu dynamicznym.

Algorytm ewolucyjny

Jest to metoda rozwiązywania problemów optymalizacyjnych wykorzystująca w tym celu mechanizmy zachodzące w naturze - w tym konkretnym przypadku mechanizmy związane z ewolucją.

Głównym podobieństwem do natury jest istnienie populacji. W przyrodzie występuje ona jako zbiór osobników tego samego gatunku o różnych cechach. W przypadku problemów optymalizacyjnych - jest to zbiór potencjalnych rozwiązań problemu. To co nazywamy tytułową ewolucją jest zjawiskiem powstawania coraz lepiej dopasowanych do środowiska gatunków, oraz wymieraniem najsłabszych z nich. Analogicznie jest w przypadku algorytmu. Każdemu z rozwiązań przypisywana jest wartość proporcjonalna do miary dopasowania rozwiązania problemu (funkcja oceny). W przypadku rozwiązań o wysokiej wartości dopasowania są one wykorzystywane (mechanizm selekcji) jako “materiał”, z którego generowane są kolejne rozwiązania różniące się od swoich “przodków”.

Sam proces generacji nowego zbioru rozwiązań również opiera się na obserwacjach natury. Wyróżniamy dwie główne metody:

- krzyżowanie - nowemu potomkowi przekazywany jest fragment rozwiązania pochodzący od obojga rodziców. Tym samym otrzymujemy nowe rozwiązanie, które będzie posiadać dobre “cechy” rozwiązań, z których powstało.
- mutacja - jedna z cech potomka zostaje losowo zmieniona. Umożliwia nam to odkrycie nowych, nieznanych dotąd “cech”, a tym samym, możemy uzyskać większa wartość funkcji oceny (jak również i mniejsza)

W celu otrzymania jak najlepszego rozwiązania proces generacji i selekcji jest powielany wielokrotnie. Ważnym aspektem, który znacząco wpływa na generowane rozwiązania jest odpowiednie wyważenie metod krzyżowania i mutacji. Pierwsza z nich skutkuje ujednoliceniem populacji rozwiązań, co może skutkować znalezieniem jedynie lokalnego optimum. Z kolei druga z nich przeciwdziała pierwszej metodzie generując nowe osobniki umożliwiając znalezienie globalnego optimum, jednakże w przypadku silnego wpływu na nowe pokolenie, może również doprowadzić do jego utracenia.

Osobnik: wektor binarny długości N: 1 oznacza wykonanie transakcji, 0 - brak transakcji
Funkcja celu: suma wartości wszystkich transakcji, jeżeli nie przekroczono wagi sprzedawanego złota, bądź 0, jeżeli przekroczono sumaryczną wagę złota we wszystkich transakcjach
Selekcja: turniejowa
Krzyżowanie: utworzenie wektora z podciągu długości k jednego osobnika i długości N - k drugiego osobnika z prawdopodobieństwem Km.
Mutacja: zmiana wartości pojedynczego bitu w wektorze osobnika z prawdopodobieństwem Pm (sprawdzenie dla każdego bitu w wektorze).

Pseudokod

- Generacja populacji startowej
- Powtarzaj do wystąpienia warunku stopu (np. wystarczająca jakość rozwiązań, ilość powtórzeń):
    - Wybierz najlepsze osobniki (selekcja).
    - Dokonaj krzyżowania pośród wybranych osobników (z określonym prawdopodobieństwem, lub przy zadanej liczebności par).
    - Dokonaj mutacji (z określonym prawdopodobieństwem).
    - Odrzuć najgorsze osobniki, zastępując je potomstwem.

Algorytm plecakowy

Niech Ai, gdzie i ∈ (1, 2, …, N) będzie wagą elementów oraz Vi, i ∈ (1, 2, …, N) - wartościami. Algorytm ma zmaksymalizować wartość elementów przy zachowaniu sumy ich wagi mniejszej bądź równej W. Niech P(i,j) będzie największą możliwą wartością, która może być otrzymana przy założeniu wagi mniejszej bądź równej j i wykorzystaniu pierwszych i elementów.

Funkcja P(i,j) definiowana jest rekurencyjnie:

- P(0, j) = 0
- P(i, 0) = 0
- P(i, j) = P(i - 1, j), jeśli Ai > j
- P(i, j) = max(P(i - 1, j), P(i - 1, j - Ai) + Vi), jeśli Ai ≤ j

Rozwiązaniem problemu jest wynik dla P(n, W).

Pseudokod:

```
for i := 0 to n do
        P[i, 0] := 0
for j := 0 to W do
        P[0, j] := 0
// rozważanie kolejno i pierwszych przedmiotów
for i := 1 to n do
        for j := 0 to W do
                //sprawdzenie czy i-ty element mieści się w plecaku o rozmiarze j
                if ( A[i] > j ) then
                        P[i, j] = P[i - 1, j]
                else
                        P[i, j] = max( P[i - 1, j], P[i - 1, j - A[i]] + V[i] )
```

Szczegóły implementacyjne

- język implementacji - Python 3.*
- system kontroli wersji - git

Plan eksperymentów

Analiza porównawcza algorytmu plecakowego oraz programowania dynamicznego z uwzględnieniem następujących kryteriów:

- czas: pobranie stempli czasowych (timestamp) w momencie uruchomienia i zakończenia programu; czas działania programu jest różnicą dwóch wartości
- złożoność pamięciowa
- złożoność obliczeniowa
- zgodność rozwiązań: porównanie wyników działania obydwu algorytmów celem sprawdzenia poprawności

Testy będą przeprowadzane w oparciu o generowane losowo tablice Ai i Vi. Obydwa algorytmy będą bazowały na tych samych danych wejściowych.