Projekt Podstawy sztucznej Inteligencji

Problem upakowania

Wojciech Zieliński, Michał Stawiarski, Łukasz Binkiewicz

# Treść projektu

Do walizki o ograniczonej pojemności V chcemy załadować przedmioty o jak największej wartości, mając wszakże na uwadze, że każdy z przedmiotów zajmuje pewną objętość. Dla podanego zbioru przedmiotów, z których każdy zajmuje pewną objętość. Dla podanego zbioru przedmiotów, z których każdy scharakteryzowany jest przez jego wartość oraz objętość, znajdź przy użyciu algorytmu ewolucyjnego zestaw przedmiotów mieszczących się w walizce, których sumaryczna wartość jest największa.

# Opis zastosowanego algorytmu

## Główny algorytm:

I. Inicjalizujemy listę przedmiotów(dane pochodzą z pliku lub są wprowadzone przez użytkownika).

-Aby odczytać dane z pliku wykorzystujemy klasę FileParser, która w konstruktorze przyjmuje nazwę pliku do parsowania, a po wywołaniu metody getItemArrayList() zwraca listę zawierającą przedmioty ze zdefiniowanymi wartościami objętości i wartości

II. Inicjalizujemy tablicę, która zawiera losowe chromosomy.

-Do realizacji tej części algorytmu używamy klasy Population, która w konstruktorze przyjmuje argument definiujący wielkość populacji, następnie używając stworzonego obiektu populacji wywołujemy funkcję generateRandomPopulation(int chrSize), która definiuje losowe genotypy chromosomów tworząc obiekty przy użyciu klasy Chromome.

III. Liczymy wartości fitnesu (wartość maksymalną plecaka) dla każdego chromosomu.

-Aby policzyć wartość fitnesu wykorzystujemy klasę Chromosome i jej metodę fitnessCalculate, która przyjmuje dwa argumenty – listę przedmiotów i pojemność plecaka.

IV. Generacja potomków.(jeśli 90% populacji ma taką samą wartość fitness)

a) Losowo wybieramy dwa chromosomy z populacji

- realizacja tej części wykonuje się w klasie KnapsackAlgorithm przy pomocy funkcji groupSelection, która przyjmuje jako argument aktualną populacji. Za pomocą tej funkcji wybieramy chromosomy, które są losowane zgodnie z założeniami wybierania grupowego, które zakłada, że lepiej przystosowane osobniki(większa wartość plecaka) wybierane są z większym prawdopodobieństwem.

b) Wykonujemy krzyżowanie jednopunktowe

-w tym kroku również korzystamy z klasy KnapsackAlgorithm i jej metody crossoverChromosomes, która jako argumenty przyjmuje populację, listę przedmiotów oraz pojemność plecaka. W metodzie dokonujemy krzyżowania losując dwa chromosomy między którymi będzie zachodziło krzyżowanie oraz locus podziału. W wyniku wykonania jednego krzyżowania dostajemy dwa nowe osobniki a cała funkcja zwraca nową populację.

c) Przeprowadzamy mutację

-mutacja przeprowadzana jest za pomocą funkcji mutateChromosomes która jako argumenty przyjmuje populacje, listę przedmiotów oraz pojemność plecaka. Funkcja losuje z zadanym prawdopodobieństwem z całej populacji jednego osobnika, który zostanie zmutowany. Bezpośrednia mutacja wykorzystuje funkcję mutateChromosome, która przyjmuje jako argument wylosowany chromosom, listę przedmiotów oraz pojemność plecaka. Po jej wykonaniu w aktualnej populacji zostaje zmutowany jeden chromosom.

V. Sprawdzamy ile generacji zostało wykonanych i porównujemy z wartością zadaną, jeśli wykonanych iteracji jest mniej niż zdefiniowaliśmy, algorytm przechodzi do punktu II.

## Algorytm wyliczania fitnessu:

I. Dla każdego genu w chromosomie, który jest ustawiony na 1 (zawiera odpowiadający przedmiot) dodaj wartość i objętość do całkowitej wartości i całkowitej objętości.

II. Sprawdzamy czy całkowita objętość jest większa od zadanej objętości, jeśli nie to przypisujemy obliczone wartości do chromosomu i kończymy algorytm dla danego chromosomu, w przeciwnym przypadku przechodzimy do następnego kroku.

III. Losowo wybieramy geny do momentu aż nie natrafimy na gen o wartości 1.

IV. Usuwamy przedmiot odpowiadający temu genowi (zerujemy wartość tego genu)

W algorytmie ważną rolę odgrywa również funkcja groupSelection, która wybiera chromosomy do krzyżowania. Ogólna idea polega na tym, że sortujemy całą populację chromosomów w porządku rosnącym i dzielimy ją na cztery przedziały równej długości( od najmniejszego dla gorszych chromosomów do największego dla chromosomów o lepszej wartości fitnesu ). Następnie wybieramy chromosomy z przedziałów od najlepszego do najgorszego odpowiednio z prawdopodobieństwem 50%, 30%, 15%, 5%.

## Opis wykorzystanych bibliotek

W projekcie została jedynie wykorzystana wbudowana biblioteka jaką jest *JavaFX.* Wykorzystaliśmy ją do stworzenie interfejsu graficznego do programu. Do budowania aplikacji używaliśmy narzędzia automatyzującego budowę programu *Maven*. Dodatkowo korzystaliśmy z *Git`a* jako narzędzia kontroli wersji.

## Instrukcja użytkowania programu

Po uruchomieniu pojawia się nam okno w którym możemy wybrać czy chcemy wprowadzać dane ręcznie czy dane zostaną pobrane z pliku.

### Wprowadzanie danych ręcznie

Pojawia się nowa scena w której wprowadzamy ilość przedmiotów, rozmiar populacji, ilość generacji oraz rozmiar plecaka. Po kliknięciu przycisku Zatwierdź w kolejnej scenie musimy wprowadzić dane objętości i wartości dla każdego przedmiotu. Po prowadzeniu danych i wciśnięciu Zatwierdź program zaczyna działać. Po skończeniu obliczeń pojawiają się dwa wykresy. Pierwszy z nich zawiera średnią wartość wszystkich plecaków populacji w danej generacji, drugi z wykresów zawiera maksymalną wartość plecaka w danej generacji.

### Wczytywanie danych z pliku

Pojawia się nowa scena w której możemy wpisać nazwę pliku z którego mają zostać pobrane dane. Po skończeniu obliczeń pojawiają się dwa wykresy. Pierwszy z nich zawiera średnią wartość wszystkich plecaków populacji w danej generacji, drugi z wykresów zawiera maksymalną wartość plecaka w danej generacji.

## Testowanie i analiza

Dla rozsądnych warunków wstępnych algorytm zachowuje się dobrze i znajduje maksymalną wartość plecaka w zadowalającej liczbie kroków. Problem pojawia się, gdy wielkość populacji jest dużo mniejsza niż ilość wszystkich kombinacji genów.

## Podział obowiązków

#### Łukasz Binkiewicz

Wstępna analiza projektu, przygotowanie materiałów.

#### Wojciech Zieliński

Interfejs graficzny

#### Michał Stawiarski

Tworzenie dokumentacji oraz debuggowanie

#### Wszyscy

Rozwijanie głównego kodu algorytmu