25.11.2019

Dokumentacja projektu

problem plecakowy

wykonali: Marcin Mozolewski i Marcin Różański

1 Treść zadania

Do plecaka o pojemności C chcemy zapakować przedmioty o jak największej wartości. Każdy przedmiot definiowany jest przez wartość W i objętość O jaką zajmuje w plecaku. Zastosować algorytm genetyczny i zamodelować pewien sposób reprezentacji genotypu do rozwiązania tego problemu. WE: plik z listą przedmiotów opisanych wartością W i objętością O, pojemność plecaka C. WY: przedmioty mieszczące się w plecaku, które maksymalizują jego wartość

2 Pełen opis funkcjonalny

2.1 Pomysł ogólny

Wykorzystany został algorytm genetyczny. Genotyp każdego osobnika zawiera x bitów, gdzie x to liczba przedmiotów, które można włożyć do plecaka. 1 w genotypie informuje, że przedmiot odpowiadający danej pozycji jest umieszczony w plecaku, a 0 informuje, że nie jest.

2.2 Funkcja celu

$$maximize\{f(x)\} = maximize\ \{\sum_{i=1}^{n} x_i w_i\}$$
$$subject\ to \sum_{i=1}^{n} x_i o_i \leq O$$
$$x_i \in \{0,1\},\ i = 1,2,...,n$$

gdzie:

- x_i = przedmiot o indeksie i
- O = maksymalna objętość plecaka
- w_i = wartość przedmiotu o indeksie i
- o_i = objętość przedmiotu o indeksie i

2.3 Algorytm

1. Wylosuj populację startową(domyślna wielkość jest równa ilości przedmiotów).
2. Metodą koła ruletki(losowanie ze zwracaniem) utwórz listę rodziców(ilość równa wielkości populacji).
3. Przeprowadź krzyżowanie jednopunktowe między sąsiadującymi ze sobą na liście rodzicami, za każdym razem tworząc 2 nowe genotypy.
4. Jeśli fenotypy kodowane przez nowo powstałe genotypy przekraczają pojemność plecaka ogranicz je (losowo neguj wartości 1 w genotypie aż warunek maksymalnej objętości nie będzie spełniony).
5. Przeprowadź na dzieciach mutację. Prawdopodobieństwo mutacji pojedynczego genu jest dobrane tak, aby około 1 na 10 genotypów był zmutowany(wartość domyślna).
6. Ponownie ogranicz dzieci.
7. Wybierz pokolenie następne, w którego skład będą wchodzić dzieci o najlepszym dopasowaniu oraz genotyp o najlepszym fenotypie z poprzedniego pokolenia(elitaryzm).
8. Korki 2-7 powtarzaj do momentu, gdy algorytm przejdzie minimalną liczbę iteracji i określona część osobników ostatniej populacji będzie miało tą samą wartość dopasowania.

Zastosowane metody selekcji rodziców, krzyżowania, mutacji i wyboru następnego pokolenia to sposoby znane z literatury, o potwierdzonej skuteczności.

3 Struktura programu

3.1 Podział odpowiedzialności

- Marcin Mozolewski: Wprowadzanie i weryfikacja danych wejściowych, generacja wyjściowego wykresu, korekty algorytmu
- Marcin Różański: Implementacja algorytmu, generacja pliku wyjściowego, dokumentacja

3.2 Podział aplikacji

W aplikacji można wyróżnić część odpowiedzialną za pobieranie danych zewnętrznych takich jak objętość plecaka, przedmioty oraz ustawienia algorytmu. Część ta również waliduje czy dane wejściowe są poprawne. Druga część programu to algorytm, w której został zaimplementowany algorytm genetyczny.

4 Pliki konfiguracyjne

4.1 plik z danymi wejściowymi

Plik w formacie txt o następującej strukturze:

1. pierwszy wiersz zawierający maksymalną objętość plecaka w postaci liczby typu Long
2. N wierszy zawierających parę wartości(objętość, wartość) oddzielonych od siebie spacją w typie Long

Przykład:
12
1 2
3 5
10 101

4.2 plik konfiguracyjny(opcjonalny)

Plik w formacie txt zawierający wiersze z własnymi ustawieniami algorytmu. W przypadku podania błędnego wiersza lub niepodania ustawienia przyjmowana jest wartość domyślna algorytmu. Klucze do konfiguracji ustawień:

1. initialPopulation - wielkość populacji(liczba naturalna)
2. chromosomePerMille -szansa na mutację chromosomu w promilach(liczba naturalna)
3. dominatorPercentage - wymagany procent tego samego dopasowania w populacji do zakończenia algorytmu(liczba naturalna)
4. iterations - minimalna liczba iteracji(liczba naturalna)
5. generateChart - czy wygenerować wykres(true/false)
6. printOldPopulations - czy wypisać wszystkie populacje(true/false)

Przykład:
generateChart = true
initialPopulation = 10
iterations = 100

5 Uruchamianie programu

5.1 Wymagania

Do uruchomienia programu w postaci jar-a wymagana jest zainstalowana Java Runtime Environment.

5.2 Uruchomienie

Aby uruchomić program należy w terminalu przejść do katalogu w którym znajduje się plik discrete-knapsack-problem-1.0.jar, a następnie w terminalu wpisać:

java -jar discrete-knapsack-problem-1.0.jar [plik wejściowy] [plik wyjściowy] [opcjonalny plik konfiguracyjny]

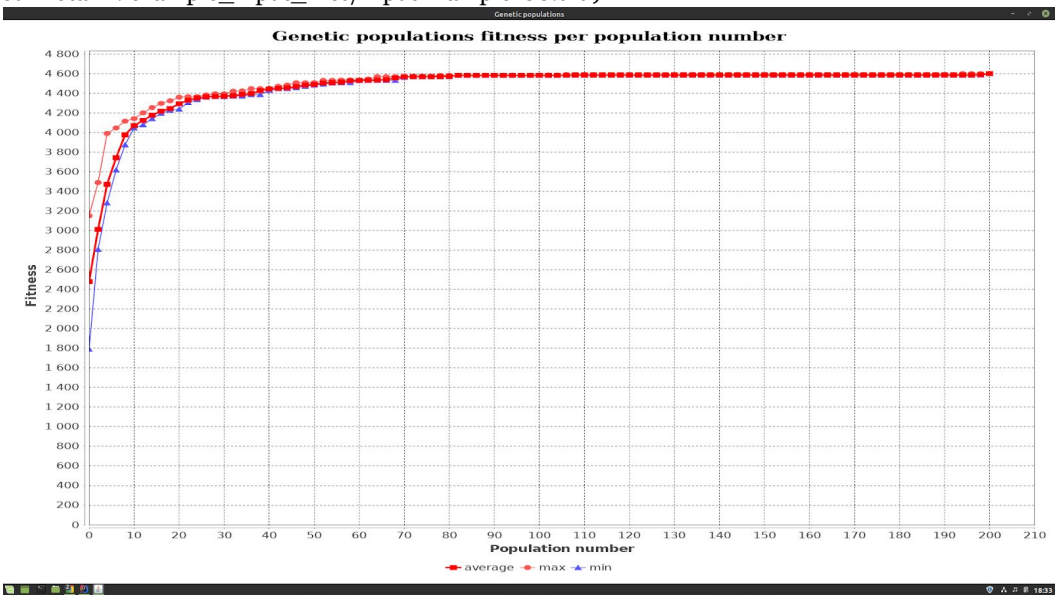
6 Przeprowadzone testy i wnioski

6.1 Testy

Dla wielkości populacji = 50, minimalnej liczby pokoleń = 100:

Ilość przed-miotów	objętość plecaka	suma wartości przedmiotów	suma objętości przed-miotów	czas działania algorytmu (s)	ustawiony wymagany % dominacji	najwyższa wartość w 1 pokoleniu	uzyskana wartość	uzyskana objętość	ilość pokoleń
100	3544	4857	4431	3,82	95	3131	4600	3535	100
1000	39444	49691	49306	58,64	80	26626	44396	39434	524
2000	78953	99442	98692	199,94	60	52611	88439	78945	925

Wykres przedstawiający przebieg programu dla danych w pierwszym wierszu tabeli (dokładny plik wejściowy z przedmiotami: example_input_files/inputExample100.txt)



6.2 Wnioski

Algorytm jest podatny na znajdowanie maksimum lokalnego zamiast całościowego. Zjawisko to jest tożsame z algorytmami genetycznymi i nie da się go całkowicie wyeliminować. Można zmniejszyć szanse zajścia go na przykład przez implementację innych metod selekcji(np. selekcji rankingowej), w których szansa na zdominowanie populacji przez 1 chromosom jest mniejsza. Wówczas wyniki otrzymane z obu metod należy porównać i wybrać ten lepszy. Lepszy wynik można uzyskać również poprzez manipulacje danymi(wielkość populacji/minimalna ilość iteracji/szansa na mutację).