Dokumentacja wstepna projektu z POP

Piotr Jabłoński (325163) i Paweł Wysocki (325248)

Listopad 2024

Contents

1 Temat projektu

Zadaniem jest rozwiązanie problemu plecakowego dla danych skorelowanych i nieskorelowanych używając algorytmu PBIL oraz porównanie z inną metodą. Należy poddać się dokładnie statycznycznej analizie wyników. Jako alternatywną metodę rozwiązania wybraliśmy algorytm A*.

2 Opis problemu

Problem plecakowy jest klasycznym problemem maksymalizacji zysku. Należy wybrać n przedmiotów z plecaka, tak aby ich sumaryczna waga ($\sum_i^n w_i$) była mniejsza niż pojemność plecaka W. Każdy z przedmiotów ma dwa parametry - waga (w) oraz wartość (p).

$$max \sum_{i=1}^{n} x_i \cdot v_i$$

$$\sum_{i=1}^{n} x_i \cdot w_i \leq W$$

gdzie $x_i \in \{0, 1\}$ - decyzja o wsadzeniu przedmiotu do pleceka

2.1 Funkcja zysku

$$f(x) = \sum_{i=1}^{n} x_i \cdot v_i,$$

Wszystkie rozwiązania produkujące sumaryczną wagę większą od W są odrzucane.

2.2 Reprezentacja rozwiązania

W klasycznym zadaniu problemu plecakowego reprezentacja zadania jest wektorem bitów, gdzie pojedynczy bit oznaczenia podjęcie decyzji o wsadzeniu przedmiotu do plecaka, może wyglądać następująco

```
01101011        wpakowanie 2,3,5,7 i 8 przedmiotu
11111111        wpakowanie wszystkich przedmiotów
00000000        nie wpakowanie żadnego przedmiotu
```

3 PBIL (Population-Based Incremental Learning)

Należy do rodziny algorytmów EDA (Estimation of Distribution Algorithm), znany jest ze swojej prostoty. Ta prostota wynika z faktu, że dystrybucja prawdopodobieństwa kolejnych bitów w chromosomie jest niezależna od punktu startowego. Dzięki temu algorytm można dowolnie modyfikować i optymalizować do konkretnego zadania.

Algorytm działa na zasadzie iteracyjnej poprawie populacji początkowej poprzez:

1. wygenerowanie M osobników z populacji P^t
2. ewualuacja i wybór najlepszych N osobników z M - podzbiór O^t
3. tworzenie nowej populacji P^{t+1} na podstawie populacji P^t oraz O^t
4. mutacja populacji P^{t+1}

Kod w pseudo-pythonie przedstawiono poniżej:

```
M, N # const

p = Population()
t = 0
while !stop:
    Pt = sample(p, M)
    Ot = select(Pt, N)
    p = update(Ot, p, a)
    p = mutate(p)
    t += 1
```

Warunek końca to może być ilość iteracji lub satysfakcjonująco dobre rozwiązanie. Algorytm należy zatrzymać gdy wektor prawdopodobieństwa się ustabilizuje tzn. gdy z iteracji do iteracji występuje bardzo mały stopień poprawy.

3.1 Reprezentacja zadania

W algorytmach ewolucyjnych rozwiązaniem zadania jest osobnik, a w każdej iteracji osobniki są mutowane indywidualnie. W przypadku algorytmów EDA sytuacja jest inna - optymalizuje się cały genotyp populacji na raz. Każda populacja jest reprezentowana jako dystrybucja prawdopodobieństwa. Algorytm PBIL należy do tej grupy i reprezentuje populację jako wektor prawdopodobieństw (p_i):

$$\boldsymbol{p}^t = [p_1^t, p_2^t, \ldots, p_n^t]$$

Ten wektor będzie optymalizowany według funkcji zysku.

3.2 Funkcja `update`

Odpowiedzialna za aktualizację wektora prawdopodobieństw bazując na ilości jedynek w $\mathbf{x}$, które znajdują się w N najlepszych rozwiązań. Wzór funkcji przedstawiono poniżej:

$$\boldsymbol{p}^{t+1} = (1-a) \cdot \boldsymbol{p}^t + a \cdot \frac{1}{N} \sum_{x \in O^t} x$$

gdzie:

- a - learning rate
- $\mathbf{x}$ - binarny wektor opisany w reprezentacji rozwiązania

4 A*

A* jest przykładem algorytmu wyczerpującego przeszukiwania przestrzeni. Jest to algorytm zupełny i optymalny, tym sensie, że jeżeli optymalne rozwiązanie istnieje, to zostanie znalezione. Jest to typowe narzędzie do rozwiązywania problemów drzewiastych, takich jak problem plecakowy. Uznaliśmy że cecha optymalności tego algorytmu pozwoli na ciekawą analizę wyników w porównaniu do alg. PBIL.

4.1 Reprezentacja rozwiązania w A*

Nieco różni się od pierwszego podejścia tym, że wektor musi zostać rozszerzony o znak ?, który oznacza **brak decyzji**. Algorytm działa na zasadzie tworzenia "ścieżki", więc tylko węzeł końcowy będzie reprezentował kompletne rozwiązanie, tzn. informacje o wpakowaniu każdego przedmiotu.

```
????????        punkt startowy algorytmu
01??????        wpakowanie 2; reszta nieznana - poziom 2 drzewa
01101???        wpakowanie 2,3,5; reszta nieznana
01101110        wpakowanie 2,3,5,6,7; węzeł końcowy
```

4.2 Funkcja zysku i heurystyczna

W przypadku problemu plecakowego A* działa na zasadzie **maksymalizacji** funkcji f, która jest definiowana przez:

$$f(x) = g(x) + h(x)$$

gdzie:

- $g(x)$ - funkcja zysku
- $h(x)$ - funkcja heurystyczna

4.2.1 Funkcja zysku

Najbardziej sensownym podejściem będzie zsumowanie wartości przedmiotów **w plecaku**

$$g(x) = \sum_{i:x_i=1}^{n} v_i$$

Jeżeli wpakujemy dodatkowy przedmiot: rozwiązanie zwiększy swoją sumaryczną wartość się o wartość wsadzonego przedmiotu.

4.2.2 Funkcja heurystyczna

Musi być:

- dopuszczalna: $g(x) + h(x) \geq g(x_t)$
- monotoniczna: $g(x_j) + h(x_j) \leq g(x_i) + h(x_i)$

Dla problemu plecakowego można użyć funkcji heurystycznej postaci:

$$h(x) = \sum_{i:x_i=?}^{n} y_i \cdot p_i$$

gdzie

- $i : x_i =?$ - indeksy w x dla przedmiotów o statusie ?
- y_i to zmienna ułamkowa, definiowana przez równość

$$\sum_{i:x_i=?}^{n} y_i \cdot w_i = W - \sum_{i=1}^{n} x_i \cdot w_i$$

5 Plan eksperymentów

Aby przeprowadzić dokładną analizę statystyczną porównującą efektywność obu metod, wymagane jest odpowiednie środowisko testowe. Do tego zadania wybraliśmy 3 różne problemy plecakowe, które różnią się od siebie poziomem skorelowania danych:

1. Dane nieskorelowane
2. Dane średnio skorelowane
3. Dane mocno skorelowane

Testy przeprowadzimy dla różnej maksymalnej ilości przedmiotów, wagi przedmiotu oraz pojemnośic plecaka. Początkowo przyjmujemy $n = 100$, $v = 10$, $W = 100$.

Takie podejście pozwoli na sprawdzenie efektywności metody PBIL w porównaniu do metody A*, które zawsze znajdzie optymalne rozwiązanie. Każdy eksperyment będzie uruchomiony wielokrotnie, a wyniki uśrednione z dokładnością do drugiego miejsca po przecinku.

Parametr, który nas interesuje podczas testowania efektywności najbardziej to **maksymalna nagrana wartość funkcji celu** oraz **ilość obliczeń funkcji celu**. Do analizy wizualnej posłużą nam wykresy, które mapują wartości funkcji celu na ilość iteracji algorytmu oraz tabelki ze średnimi wartościami i standardowym odchyleniem. Informacje, które będziemy zbierać:

```
{
        maksymalna_wartość_funkcji_celu,        # najlepszego osobnika
        końcowa_wartość_funkcji_celu,           #
        średnia_wartość_funkcji_celu,           # całej populacji
        odchylenie_standardowe_funkcji_celu     #
}
```