AMHE - Sprawozdanie wstępne

Bartosz Latosek,
Mateusz Krakowski

Kwiecień 2025

Spis treści

1 Temat

*Rozwiązać problem plecakowy dla danych skorelowanych i nieskorelowanych używając algorytmu **PBIL** (Population-Based Incremental Learning), porównując z wybraną metaheurystyką. Wymagana dokładną analiza statystyczna przedstawionych wyników. Zalecane konsultacje u prowadzącego przed przystąpieniem do pracy nad projektem.*

2 Opis problemu

Problem plecakowy (znany również jako problem złodzieja) jest klasycznym zagadnieniem optymalizacyjnym. W problemie tym mamy do czynienia ze zbiorem n przedmiotów, gdzie każdy i-ty przedmiot posiada:

- wartość $v_i \in R^+$
- ciężar $w_i \in R^+$

Problem polega na znalezieniu wektora decyzyjnego $x = (x_1, x_2, \ldots, x_n)$, gdzie $x_i \in \{0, 1\}$, który maksymalizuje łączną wartość zabranych przedmiotów, nie przekraczając pojemności plecaka $C \in R^+$:

$$\text{Max} \sum_{i=1}^{n} v_i x_i \tag{1}$$

przy ograniczeniu:

$$\sum_{i=1}^{n} w_i x_i \leq C \tag{2}$$

gdzie:

- $x_i = 1$ oznacza, że przedmiot i-ty został spakowany
- $x_i = 0$ oznacza, że przedmiot i-ty został pominięty

Powyższy opis dotyczy tzw. wariancji dyskretnej problemu plecakowego, w której decyzja o umieszczeniu przedmiotu w plecaku jest binarna - albo umieszczamy cały przedmiot, albo nie. W literaturze można również znaleźć ciągłą wersję tego problemu, w której dopuszczalne jest umieszczenie części przedmiotu w plecaku, ale na potrzeby niniejszego projektu i ograniczeń, jakie nakłada wykorzystanie algorytmu PBIL można założyć, że analizowany będzie wyłącznie dyskretny wariant tego problemu.

3 Cel projektu

Celem projektu jest porównanie statystyczne działania dwóch algorytmów rozwiązujących zdefiniowany problem plecakowy. Pierwszym z analizowanych algorytmów będzie narzucony odgórnie algorytm **PBIL** (opisany w dalszych rozdziałach). Jako, że jest to odmiana algorytmu ewolucyjnego, jako algorytm konkurencyjny wybrany został **klasyczny algorytm ewolucyjny** (również opisany w późniejszych rozdziałach).

4 Dane

W niniejszym projekcie zdecydowano stworzyć i wykorzystać własny generator przedmiotów do problemu plecakowego, co związane było z późniejszym ułatwieniem analizy wniosków i badań statystycznych. Wykorzystanie funkcji generatora umożliwia parametryzację wykorzystanych danych pod kątem:

- **Wielkości zbioru danych**, przy czym n oznacza liczbę przedmiotów:
  - Małe - $n \leq 10$
  - Średnie - $10 < n \leq 50$
  - Duże - $n > 50$
- **Skorelowania danych** pochodzących ze zbioru dostępnych przedmiotów:
  - **Skorelowane** - wartości v_i i w_i generowane z rozkładu normalnego o zadanych parametrach:
$$w_i \sim \mathcal{N}(\mu_w, \sigma_w^2), \quad v_i \sim \mathcal{N}(\mu_v, \sigma_v^2) \tag{3}$$
  - **Nieskorelowane** - generowanie niezależnych wartości v_i i w_i z rozkładu jednostajnego:
$$v_i \sim U(v_{\min}, v_{\max}), \quad w_i \sim U(w_{\min}, w_{\max}) \tag{4}$$

5 Opis algorytmów

5.1 PBIL - Population-Based Incremental Learning

Algorytm PBIL (Population-Based Incremental Learning) to połączenie algorytmów genetycznych z uczeniem prawdopodobieństwa. Operuje na wektorze prawdopodobieństwa P:

$$p_l(x) = (p_l(x_1), p_l(x_2), \ldots, p_l(x_n))$$

, gdzie $p_l(x_i)$ określa prawdopodobieństwo wystąpienia wartości 1 na $i - tej$ pozycji genomu w generacji l.

Początkowy genom populacji inicjalizowany jest z wartością $p_1(x) = 0.5$, co gwarantuje generację losowych osobników do momentu optymalizacji genomu. W każdej iteracji algorytmu generowane jest M osobników populacji (wszystkie oparte na głównym genomie prawdopodobieństwa), z których następnie wybierane jest N najlepszych rozwiązań. Następnie, najlepsze osobniki są wykorzystywane do aktualizacji genomu prawdopodobieństwa zgodnie z regułą inspirowaną *Hebbem*:

$$p_{l+1}(x) = (1 - \alpha)p_l(x) + \alpha \frac{1}{N} \sum_{K=1}^{N} x_{k:M}^l$$

, gdzie $\alpha \in (0, 1]$ to prędkość uczenia (*ang. learning rate)*.

Po każdej aktualizacji genomu prawdopodobieństwa, generowane są nowe osobniki populacji i cykl powtarza się do momentu osiągnięcia warunku stopu (np. przekroczenia maks. liczby epok). W celu podsumowania przebiegu algorytmu został stworzony niniejszy pseudokod:

Algorithm 1: Population-Based Incremental Learning (PBIL)

Initialize probability vector P with 0.5 for all positions
while $l <= MAX\_EPOCHS$ **do**
  Generate M individuals based on P:
  foreach *individual* **do**
    foreach *gene position* i **do**
      Set gene to 1 with probability $P[i]$, otherwise set to 0
    end
  end
  Evaluate fitness of all M individuals
  Select the N best individuals based on fitness
  foreach *gene position* i **do**
    Compute mean value of gene i in the N best individuals

$$\text{mean\_value} = \frac{1}{N} \sum_{K=1}^{N} x^{l}_{k:M}$$

    Update $P[i]$ using:

$$P[i] \leftarrow (1 - \alpha)P[i] + \alpha \cdot \text{mean\_value}$$

  end
end
Return optimized probability vector P

5.2 Klasyczny algorytm ewolucyjny

Klasyczny algorytm ewolucyjny jest dobrze znanym podejściem optymalizacyjnym, szeroko stosowanym w różnych dziedzinach. W związku z tym, że jego szczegóły są dość powszechne, w ramach przypomnienia został zamieszczony poniższy pseudokod:

Algorithm 2: Classical Evolutionary Algorithm

Initialize population of size M with random individuals
while $l \leq MAX\_EPOCHS$ **do**
  Evaluate fitness of all M individuals
  Select N best individuals based on fitness (elitism)
  Generate offspring through crossover: **foreach** *pair of selected parents* **do**
    Perform crossover to create new offspring
  end
  Apply mutation: **foreach** *individual in offspring* **do**
    foreach *gene position* i **do**
      With probability p_{mut}, flip the gene value
    end
  end
  Replace the population with selected N best solutions + generated offspring
end
Return best solution found in population

- W ramach **krzyżowania** najlepszych osobników wykorzystane zostanie krzyżowanie z wykorzystaniem punktu przecięcia (stanowiącego środek genomu), zgodnie z poniższym przykładem:

  Rodzic 1: 1011 | 0010

  Rodzic 2: 0110 | 1101

  Punkt podziału: 4-ty bit

  Potomek 1: 1011 | 1101

  Potomek 2: 0110 | 0010

- Mutacje stanowić będzie zamiana bitu na $i - tej$ pozycji genomu potomstwa.

  Osobnik: 10**1**10010

  Mutacja na pozycji i=3

  Zmutowany Osobnik: 10**0**10010

6 Opis eksperymentów

Celem eksperymentu jest porównanie efektywności algorytmu **PBIL** i **klasycznego algorytmu ewolucyjnego** w kontekście problemu plecakowego. Badanie będzie obejmowało zarówno aspekty jakościowe (np. najlepsze rozwiązania), jak i ilościowe (np. czas obliczeń, wykorzystanie pamięci). W szczególności, analizowane będą:

- **Jakość rozwiązań**: średnia jakość rozwiązania w różnych iteracjach algorytmu, najlepszy oraz najgorszy wynik uzyskany w każdej z rund eksperymentalnych.
- **Czas obliczeń**: czas wykonania obydwu algorytmów w zależności od liczby przedmiotów w problemie plecakowym.
- **Zajętość pamięci**: zużycie pamięci w zależności od wielkości problemu.
- **Złożoność czasowa**: wykazanie złożoności problemu przez analizę wzrostu zużycia czasu i pamięci w zależności od wielkości zbiorów danych

6.1 Procedura eksperymentalna

1. **Inicjalizacja**:
   - Inicjalizacja obu algorytmów (PBIL oraz klasycznego algorytmu ewolucyjnego).
   - Ustawienie takich samych parametrów dla obu algorytmów (liczba epok, parametry zbioru danych...)
2. **Powtarzanie eksperymentu**:
   - Każdy eksperyment będzie powtarzany dla różnych rozmiarów problemu (mały, średni i duży).
   - Każdy eksperyment zostanie powtórzony T razy (np. 100 razy), aby uzyskać wiarygodne dane statystyczne.
3. **Zbieranie wyników**:
   - Dla każdego przebiegu algorytmu zebrane będą następujące dane:
     - **Czas obliczeń** (w sekundach)

- **Wykorzystanie pamięci** (w MB)
- **Średnia jakość rozwiązania** (średnia wartość z najlepszych rozwiązań po każdej iteracji)
- **Najlepszy i najgorszy wynik** (najlepsze i najgorsze rozwiązanie uzyskane w danym eksperymencie)

6.2 Analiza statystyczna wyników

- **Średnia i odchylenie standardowe**: Obliczymy średnią i odchylenie standardowe dla jakości rozwiązań uzyskanych przez oba algorytmy. Średnia poda nam ogólną tendencję, a odchylenie standardowe pomoże ocenić zmienność wyników.

$$\text{Średnia} = \frac{1}{T}\sum_{i=1}^{T} x_i$$

$$\text{Odchylenie standardowe} = \sqrt{\frac{1}{T}\sum_{i=1}^{T}(x_i - \text{średnia})^2}$$

- **Test statystyczny - Test t-Studenta**: Porównanie średnich wyników algorytmów PBIL i klasycznego algorytmu ewolucyjnego. Test t-Studenta zostanie wykorzystany, aby ocenić, czy różnice w jakości rozwiązań są statystycznie istotne.
- **Analiza wyników**: Porównanie wyników obu algorytmów na różnych instancjach problemu plecakowego (różna liczba przedmiotów n). Obliczenie najlepszego i najgorszego wyniku dla obu algorytmów. Analiza różnicy w jakości rozwiązań między algorytmami.

6.3 Pomiar złożoności czasowej i pamięciowej

- **Złożoność czasowa**: Mierzymy czas obliczeń algorytmów na różnych instancjach problemu plecakowego o różnych rozmiarach (mały, średni i duży). Zbadamy, jak czas obliczeń rośnie w miarę wzrostu liczby przedmiotów. Oczekujemy, że złożoność czasowa obu algorytmów będzie rosła wykładniczo.
- **Zajętość pamięci**: Będziemy rejestrować użycie pamięci przez oba algorytmy w trakcie obliczeń, aby ocenić różnice w wymaganiach pamięciowych.

6.4 Podsumowanie analizy

Po przeprowadzeniu eksperymentów i analizie statystycznej wyników, można będzie określić, który algorytm jest bardziej efektywny pod względem jakości rozwiązań, czasu obliczeń oraz wykorzystania pamięci. Porównanie obu algorytmów na różnych instancjach problemu plecakowego pozwoli również na oszacowanie złożoności czasowej i oceny, jak zwiększająca się liczba przedmiotów wpływa na wydajność algorytmów.