**Politechnika Warszawska**

Wydział Elektroniki i Technik Informacyjnych

Algorytmy Ewolucyjne

**Zadanie projektowe #2**

Wykonawca zadania

Małkowski Mateusz

Prowadzący projekt

dr inż. Grzegorz Bogdan

Małkowski Mateusz

Numer indeksu: 321358

Numer konta USOS: 1172008

Adres email: 01172008@pw.edu.pl

Spis treści

[Cel zadania 3](#_Toc167316220)

[Kryteria doboru optymalnych parametrów 3](#_Toc167316221)

[Warunek zatrzymania 3](#_Toc167316222)

[Selekcja turniejowa 3](#_Toc167316223)

[Funkcja celu z karą 3](#_Toc167316224)

[Szansa mutacji 4](#_Toc167316225)

[Krzyżowanie 4](#_Toc167316226)

[Wyniki 5](#_Toc167316227)

[Listing programów 9](#_Toc167316228)

# Cel zadania

Celem zadania było znalezienie rozwiązania problemu plecakowego za pomocą algorytmu genetycznego. Problem plecakowy polega na wybraniu zbioru przedmiotów o określonych wagach i wartościach, tak aby maksymalizować sumaryczną wartość przedmiotów w plecaku, przy jednoczesnym nieprzekroczeniu jego maksymalnej pojemności wagowej.

Parametry Algorytmu

* Liczba przedmiotów: 32
* Wagi przedmiotów: Losowane z rozkładu równomiernego z przedziału [0.1, 1.0] z dokładnością do 0.1
* Wartości przedmiotów: Losowane z rozkładu równomiernego z przedziału [1, 100] z dokładnością do 1
* Maksymalna waga plecaka: 30% sumy wag wszystkich przedmiotów

# Kryteria doboru optymalnych parametrów

## Warunek zatrzymania

Za warunek zatrzymania algorytmu wybrano brak poprawy wartości funkcji celu przez 50 generacji z progiem poprawy . Pozwala on na wyjście algorytmu z minimów lokalnych i daje więcej czasu na znalezienie lepszego rozwiązania. Jest bardzo uniwersalny i zapewnia dobre wyważenie między efektywnością czasową a jakością znalezionego rozwiązania. Jednakże, odpowiednie dostrojenie parametrów tych kryteriów jest kluczowe dla optymalnego działania algorytmu i może wymagać dostosowania w zależności od specyfiki problemu. Jego wadą jest potencjalna nieefektywność w sytuacjach, gdy poprawa jest bardzo powolna.

## Selekcja turniejowa

Selekcja turniejowa pozwala na zachowanie różnorodności dzięki losowości jednocześnie będąc wydajna obliczeniowo. Pozwala na łatwe dostosowanie presji selekcyjnej poprzez zmianę rozmiaru turnieju i dobrze zachowuje różnorodność genetyczną w populacji. Jednakże, zbyt duża presja selekcyjna i losowość mogą prowadzić do problemów z przedwczesną konwergencją i niestabilnością wyników.

## Funkcja celu z karą

Zastosowano funkcję celu z karą co zapewnia, że rozwiązania przekraczające maksymalną wagę są penalizowane, co skutecznie prowadzi algorytm do znajdowania rozwiązań zgodnych z ograniczeniami problemu. Dzięki jej zastosowaniu algorytm genetyczny może lepiej różnicować pomiędzy rozwiązaniami, które nie spełniają warunków. Nawet jeśli rozwiązanie ma wysoką wartość, jeżeli przekracza ono maksymalną wagę, kara sprawia, że jego fitness jest niższy. Dzięki temu algorytm może skoncentrować swoje wysiłki na bardziej obiecujących obszarach przestrzeni rozwiązań. Jej wadą jest konieczność obliczenia kary dla każdego rozwiązania co lekko zwiększa złożoność obliczeniową.

## Szansa mutacji

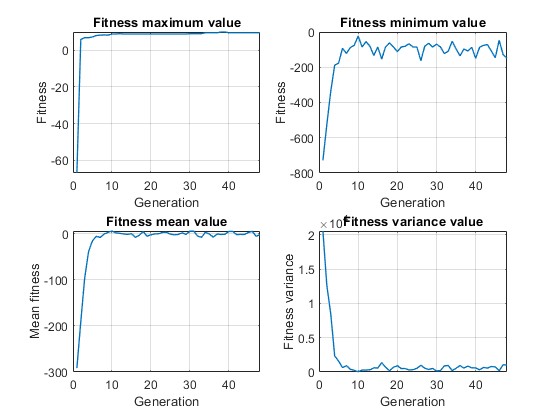
Najlepsza szansa mutacji dla stworzonego algorytmu wynosi 0.01. Przetestowano algorytm z różnymi jej wartościami i tylko dla niej algorytm nie zatrzymywał się w minimach lokalnych. Z wykresów przedstawionych na rysunkach 2, 3, 4 widać, że dla większego prawdopodobieństwa mutacji wartość średnia i wariancja nie zbliżały się do zera i miały większy rozstrzał.

Z tabel 2,3,4 widać, że dla szansy mutacji = 0.01 znaleziono największą wartość najlepszego rozwiązania. Dla większych jej wartości algorytm nie zachowywał wystarczająco dużo pozytywnych cech rodziców, by dojść do lepszych wyników. Przez zatrzymanie się algorytmu w minimum lokalnym dla większych szans mutacji algorytm kończył pracę po mniejszej ilości iteracji.

## Krzyżowanie

Algorytm przetestowano z krzyżowaniem dwupunktowym i jednopunktowym. Każdorazowo algorytm miał 95% szansy na krzyżowanie i 5% na zostawienie potomków elitarnych, co pozwoliło na stworzenie zróżnicowanego potomstwa i zwiększenie znaczenia elity w populacji. Z tabeli 1 i 2 widać, że dla jednopunktowego algorytm znalazł większą wartość najlepszego rozwiązania. Jest też mniej wymagające obliczeniowo niż krzyżowanie dwupunktowe. Z rysunków 1 i 2 przedstawiających wykresy wartości funkcji dla obu krzyżowań widać, że rodzaj krzyżowania nie miał dużego wpływy na wartość średnią i wariancję.

# Wyniki



Rysunek 1:Wykresy wartości funkcji celu dla krzyżowania dwupunktowego z szansą mutacji 0.01

|  |  |
| --- | --- |
| Szansa mutacji = 0.01 | |
| Typ krzyzowania dwupunktowy | |
| Wartosc najlepszego rozwiazania | 9.8 |
| Waga najlepszego rozwiazania | 446 |
| Srednia liczba potomkow krzyzowania | 47,6 |
| Srednia liczba potomkow elitarnych | 2,4 |
| Srednia liczba potomkow zmutowanych | 14 |
| Liczba iteracji | 88 |

Tabela 1:Wyniki dla krzyżowania dwupunktowego z szansą mutacji 0.01

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | 0 | 1 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 1 |

Tabela 2:Najlepszy wynik dla krzyżowania dwupunktowego z szansą mutacji 0.01

Obraz zawierający tekst, diagram, linia, Wykres

Opis wygenerowany automatycznie

Rysunek 2:Wykresy wartości funkcji celu dla krzyżowania jednopunktowego z szansą mutacji 0.01

|  |  |
| --- | --- |
| Szansa mutacji = 0.01 | |
| krzyżowanie jednopunktowe | |
| Wartosc najlepszego rozwiazania | 10.1 |
| Waga najlepszego rozwiazania | 446 |
| Srednia liczba potomkow krzyzowania | 47,7 |
| Srednia liczba potomkow elitarnych | 2,3 |
| Srednia liczba potomkow zmutowanych | 13,5 |
| Liczba iteracji | 117 |

Tabela 3:Wyniki dla krzyżowania jednopunktowego z szansą mutacji 0.01

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | 1 | 1 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 1 |

Tabela 4:Najlepszy wynik dla krzyżowania jednopunktowego z szansą mutacji 0.01

Obraz zawierający tekst, diagram, linia, Wykres

Opis wygenerowany automatycznie

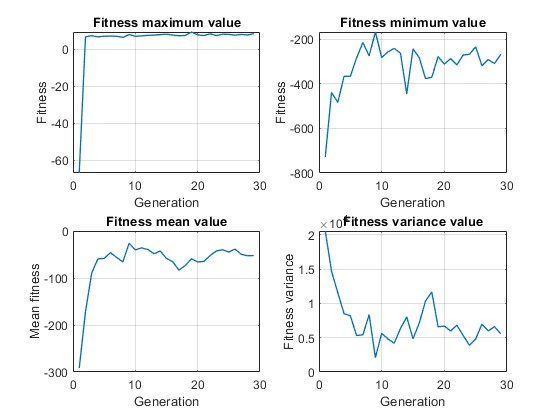
Rysunek 3:Wykresy wartości funkcji celu dla krzyżowania jednopunktowego z szansą mutacji 0.05

|  |  |
| --- | --- |
| krzyżowanie jednopunktowe | |
| Szansa mutacji | 0,05 |
| Wartosc najlepszego rozwiazania | 8.4 |
| Waga najlepszego rozwiazania | 432 |
| Srednia liczba potomkow krzyzowania | 48 |
| Srednia liczba potomkow elitarnych | 2 |
| Srednia liczba potomkow zmutowanych | 39,8 |
| Liczba iteracji | 73 |

Tabela 5:Wyniki dla krzyżowania jednopunktowego z szansą mutacji 0.05

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | 0 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 0 | 1 | 1 | 1 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |

Tabela 6:Najlepszy wynik dla krzyżowania jednopunktowego z szansą mutacji 0.05



Rysunek 4:Wykresy wartości funkcji celu dla krzyżowania jednopunktowego z szansą mutacji 0.1

|  |  |
| --- | --- |
| krzyżowanie jednopunktowe | |
| Szansa mutacji | 0,1 |
| Wartosc najlepszego rozwiazania | 8 |
| Waga najlepszego rozwiazania | 418 |
| Srednia liczba potomkow krzyzowania | 48,1 |
| Srednia liczba potomkow elitarnych | 1,9 |
| Srednia liczba potomkow zmutowanych | 47,7 |
| Liczba iteracji | 69 |

Tabela 7:Wyniki dla krzyżowania jednopunktowego z szansą mutacji 0.1

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 1 | 1 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 1 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 1 |

Tabela 8:Najlepszy wynik dla krzyżowania jednopunktowego z szansą mutacji 0.1

# Listing programów

close all;

numer\_albumu=321358; %% Wpisz swój numer albumu

rng(numer\_albumu);

N=32;

items(:,1)=round(0.1+0.9\*rand(N,1),1);

items(:,2)=round(1+99\*rand(N,1));

max\_weight = 0.3\*sum(items(:,2));

test\_algorythm(items, max\_weight, 'one point', 0.01);

% function used to test the algotythm

function test\_algorythm(items, max\_weight, crossover\_type, mutation\_chance)

[best\_solution, fitness\_history, offspring\_count] = genetic\_knapsack(items, max\_weight , mutation\_chance, crossover\_type);

disp(['Mutation chance: ', num2str(mutation\_chance)])

disp(['Crossover type: ', crossover\_type])

disp(['Best solution:', num2str(best\_solution)]);

disp(['Best solution value: ', num2str(best\_solution\*items(:,1))]);

disp(['Best solutin weight: ', num2str(best\_solution\*items(:,2))]);

disp(['mean crossover offspring: ', num2str(mean(offspring\_count(:,1)))])

disp(['mean elite offspring: ', num2str(mean(offspring\_count(:,2)))])

disp(['mean mutated offspring: ', num2str(mean(offspring\_count(:,3)))])

plot\_fitness(fitness\_history)

end

% genetic algorythm

function [best\_solution, fitness\_history, offspring\_count] = genetic\_knapsack(items, max\_weight, mutation\_chance, crossover\_type)

% parameters

population\_size = 50;

num\_generations = 10000;

crossover\_rate = 0.95;

patience = 50;

improvement\_threshold = 1e-3;

% initialization

population = randi([0, 1], population\_size, size(items, 1));

fitness\_history = zeros(num\_generations, 5);

offspring\_count = zeros(num\_generations, 3);

best\_fitness = -Inf;

generations\_without\_improvement = 0;

for generation = 1:num\_generations

fitness = evaluate(population, items, max\_weight);

[current\_best\_fitness, ~] = max(fitness);

fitness\_history(generation, 1) = current\_best\_fitness;

fitness\_history(generation, 2) = min(fitness);

fitness\_history(generation, 3) = mean(fitness);

fitness\_history(generation, 4) = var(fitness);

% stopping

if current\_best\_fitness > best\_fitness + improvement\_threshold

best\_fitness = current\_best\_fitness;

generations\_without\_improvement = 0;

else

generations\_without\_improvement = generations\_without\_improvement + 1;

end

if generations\_without\_improvement >= patience

fitness\_history = fitness\_history(1:(generation-patience+10), 1:4);

offspring\_count = offspring\_count(1:(generation-patience+10), 1:3);

break;

end

offspring\_count(generation, 1) = 0;

offspring\_count(generation, 2) = 0;

offspring\_count(generation, 3) = 0;

% Selection

selected\_indices = tournament\_selection(fitness, population\_size);

selected\_population = population(selected\_indices, :);

% crossover

if crossover\_type == 'two point'

% two point crossover

new\_population = zeros(size(population));

for i = 1:2:population\_size

parent1 = selected\_population(randi([1, population\_size]), :);

parent2 = selected\_population(randi([1, population\_size]), :);

if rand < crossover\_rate

points = sort(randperm(size(items, 1), 2));

new\_population(i, :) = [parent1(1:points(1)), parent2(points(1)+1:points(2)), parent1(points(2)+1:end)];

new\_population(i + 1, :) = [parent2(1:points(1)), parent1(points(1)+1:points(2)), parent2(points(2)+1:end)];

offspring\_count(generation, 1) = offspring\_count(generation, 1) + 2;

else % 20% szansa na brak krzyżowania (kopiowanie rodziców)

new\_population(i, :) = parent1;

new\_population(i + 1, :) = parent2;

offspring\_count(generation, 2) = offspring\_count(generation, 2) + 2;

end

end

else

% one point crossover

new\_population = zeros(size(population));

for i = 1:2:population\_size

parent1 = selected\_population(randi([1, population\_size]), :);

parent2 = selected\_population(randi([1, population\_size]), :);

if rand < crossover\_rate

point = randi([1, size(items, 1)-1]);

new\_population(i, :) = [parent1(1:point) parent2(point+1:end)];

new\_population(i+1, :) = [parent2(1:point) parent1(point+1:end)];

offspring\_count(generation, 1) = offspring\_count(generation, 1) + 2;

else

new\_population(i, :) = parent1;

new\_population(i+1, :) = parent2;

offspring\_count(generation, 2) = offspring\_count(generation, 2) + 2;

end

end

end

% Mutation

for i = 1:population\_size

mutated = false;

for j = 1:size(items, 1)

if rand < mutation\_chance

new\_population(i, j) = ~new\_population(i, j);

mutated = true;

end

end

if mutated

offspring\_count(generation, 3) = offspring\_count(generation, 3) + 1;

end

end

population = new\_population;

end

fitness = evaluate(population, items, max\_weight);

[~, best\_index] = max(fitness);

best\_solution = population(best\_index, :);

end

% function generating plots

function plot\_fitness(fitness\_history)

figure;

subplot(2,2,1)

plot(fitness\_history(:,1), 'Linewidth', 1)

xlabel('Generation');

ylabel('Fitness');

title('Fitness maximum value')

grid on;

subplot(2,2,2)

plot(fitness\_history(:,2), 'Linewidth', 1)

xlabel('Generation');

ylabel('Fitness');

title('Fitness minimum value')

grid on;

subplot(2,2,3)

plot(fitness\_history(:,3), 'Linewidth', 1)

xlabel('Generation');

ylabel('Mean fitness');

title('Fitness mean value')

grid on;

subplot(2,2,4)

plot(fitness\_history(:,4), 'Linewidth', 1)

xlabel('Generation');

ylabel('Fitness variance');

title('Fitness variance value')

grid on;

end

% fitness function

function fitness = evaluate(population, items, max\_weight)

values = population \* items(:, 1);

weights = population \* items(:, 2);

penalty = max(0, weights - max\_weight);

fitness = values - penalty;

end

% tournament selection function

function selected\_indices = tournament\_selection(fitness, population\_size)

tournament\_size = 3;

selected\_indices = zeros(population\_size, 1);

for i = 1:population\_size

competitors = randi([1, population\_size], tournament\_size, 1);

[~, best] = max(fitness(competitors));

selected\_indices(i) = competitors(best);

end

end