Dokumentacja projektu 2 NAI

Algorytm genetyczny oraz symulowanie wyżarzania dla problemu NP-trudnego –   
„Dyskretny Problem Plecakowy”

Autor: Bartosz Kalinowski s11027, Data: 2015-01-31

*Źródła:* [*https://github.com/kelostrada/genetic-knapsack*](https://github.com/kelostrada/genetic-knapsack)

Spis treści

[1. Krótki opis rozważanego problemu 3](#_Toc410489310)

[2. Opis rozwiązań technicznych 3](#_Toc410489311)

[2.1 Reprezentacja zbioru przedmiotów 3](#_Toc410489312)

[2.2 Reprezentacja rozwiązań (Kodowanie rozwiązań w strukturze chromosomu) 3](#_Toc410489313)

[2.3 Relacja sąsiedztwa 3](#_Toc410489314)

[2.4 Dane testowe 3](#_Toc410489315)

[3. Opis rozwiązań zastosowanych w algorytmach 4](#_Toc410489316)

[3.1 Opis algorytmu genetycznego 4](#_Toc410489317)

[3.2 Opis algorytmu wyżarzania 4](#_Toc410489318)

[3.3 Operacja krzyżowania – Crossover 4](#_Toc410489319)

[3.4 Operacja mutacji – Mutation 4](#_Toc410489320)

[3.5 Metoda selekcji 4](#_Toc410489321)

[3.6 Funkcja wyżarzania 4](#_Toc410489322)

# Krótki opis rozważanego problemu

Rozważany jest problem „plecakowy”. Jest to problem NP-trudny, dlatego też skorzystamy z rozwiązań genetycznych. Wariant problemu jaki autor próbuje zbadać to najprostszy z możliwych wariant z jednym plecakiem który posiada określoną wytrzymałość (maksymalną wagę przedmiotów które może przechować). Zadany jest także zbiór przedmiotów, każdy z określoną wartością i wagą które próbujemy schować do plecaka. Przedmioty oczywiście raczej nie powinny dać się schować wszystkie na raz do plecaka i problemem jest wybranie takiego układu przedmiotów, żeby schować do plecaka przedmioty o jak największej wartości, przy tym nie przekraczając zadanego limitu wagowego.

Formalnie problem może być zdefiniowany:

Mamy do dys­pozycji plecak o maksymalnej pojemności Boraz zbiór Nelementów \{x_1, x_j, ..., x_N\}, przy czym każdy element ma określoną wartość c_{j}oraz wielkość w_{j}.

\sum_{j=1}^N c_j x_j.

Zmaksymalizuj:

\sum_{j=1}^N w_j x_j \le B, \quad \quad x_j = 0\;\mbox{lub}\;1, \quad j=1,\dots,n.

przy założeniach:

# Opis rozwiązań technicznych

## Reprezentacja zbioru przedmiotów

Zbiór przedmiotów chowanych do plecaka przechowywany jest w formie tablicy ze strukturami posiadającymi dwa pola – wartość i waga. Kiedy któreś z rozwiązań potrzebuje sprawdzić czy jest poprawne i czy przedmioty mieszczą się do plecaka musi odwołać się do tej tablicy.

## Reprezentacja rozwiązań (Kodowanie rozwiązań w strukturze chromosomu)

Autor reprezentuje rozwiązania przy pomocy wektora binarnego. Każda jedynka na wektorze odpowiada informacji, że dany przedmiot został wybrany do bycia włożonym do plecaka.

Przykładowy wektor: **(1,0,0,1,1)**

Taki wektor oznacza, że wybraliśmy przedmioty z indeksów 0, 3, 4 i zostały one włożone do plecaka.

## Relacja sąsiedztwa

W przypadku algorytmów optymalizacyjnych takich jak właśnie rozważany algorytm symulowanego wyżarzania potrzebujemy zdefiniować relacje sąsiedztwa pomiędzy układami rozwiązań. W rozważanym problemie zastosowana została relacja sąsiedztwa która bazuje na mechanizmie mutacji z implementacji klasycznego algorytmu genetycznego (opis dalej). Najprościej jednak ujmując dany wektor jest w sąsiedztwie ze wszystkim wektorami które można utworzyć poprzez zastosowanie na nim jednej mutacji.

## Dane testowe

Zbiór danych testowych do został wygenerowany losowo. Ten sam zbiór populacji wykorzystany zostanie przy algorytmie genetycznym co przy algorytmie symulowanego wyżarzania.

# Opis rozwiązań zastosowanych w algorytmach

## Opis algorytmu genetycznego

Użyty algorytm składa się z kilku prostych kroków.

1. Generujemy początkową losową populację rozwiązań
2. Wybieramy dwa najlepsze rozwiązania z populacji i zapisujemy do nowego zbioru populacji
3. Losujemy dwukrotną ilość par rozwiązań ile jest elementów w podstawowej populacji
4. Stosujemy krzyżówkę między tymi parami i generujemy dwójkę dzieci
5. Jeżeli zostanie spełniony warunek losowy (domyślnie 5%), to zostanie zastosowana mutacja na dzieciach wynikających z krzyżówki
6. Jeżeli dzieci spełniają warunki (są rozwiązaniami) i nie ma ich w zbiorze nowej populacji to dokładamy je do nowej populacji
7. Sortujemy nową populację względem najlepszych rozwiązań i wyciągamy tyle ile było w podstawowej populacji
8. Nadpisujemy starą populację nową i jeśli najlepsze rozwiązanie nie powtórzyło się n razy (domyślnie 100) to wracamy do kroku 2.

## Opis algorytmu wyżarzania

Algorytm wyżarzania opiera się na idei, że wraz z upływającym czasem temperatura układu coraz bardziej wygasa i dzięki temu im jest późniejszy etap algorytmu tym mniejsza szansa żeby wejść na nową ścieżkę. Jeśli chodzi o ścieżkę to chodzi o ścieżkę w kontekście algorytmu wspinaczkowego.

## Operacja krzyżowania – Crossover

Zastosowana została standardowa operacja krzyżowania dla klasycznego algorytmu genetycznego. Tzn podczas krzyżowania następuje losowanie indeksu i złączenie dwóch wektorów na tym indeksie. Tzn dostajemy dwa nowe wektory będące dziećmi poprzednich, mające jedną część jednego i drugą drugiego wektora.

## Operacja mutacji – Mutation

Zastosowana operacja mutacji jest bardzo prosta. Losujemy dwa indeksy w wektorze rozwiązań i zamieniamy na nich wartości. Nic więcej się nie dzieje, także czasami mutacja może nawet nie wprowadzić żadnych zmian. Taka interpretacja mutacji oznacza po prostu wyciągnięcie jednego przedmiotu z plecaka i włożenie innego.

## Metoda selekcji

Rozwiązania porównywane są w bardzo prosty sposób. Sumowana jest wartość każdego przedmiotu który został włożony do plecaka. Jeżeli maksymalny rozmiar plecaka zostanie przekroczony (jeśli chodzi o wagi) to wartość podawana jest 0. Jeżeli nie przekroczymy rozmiaru plecaka to suma wartości jest wynikiem tej funkcji. Potem na podstawie tego wyniku porównujemy różne rozwiązania w celu sprawdzenia które rozwiązanie jest „lepsze”.

## Funkcja wyżarzania

http://www4a.wolframalpha.com/Calculate/MSP/MSP50741g0023ae8334499300003gg97681id1046a3?MSPStoreType=image/gif&s=58&w=66.&h=36.

Zastosowana funkcja wyżarzania opiera się na prostej funkcji , gdzie T0 to temperatura początkowa.