Kierunek: Informatyka, semestr V
Specjalność: Aplikacje Internetowe i Mobilne
Rok akademicki: 2022/2023

# Inteligencja obliczeniowa
## LABORATORIUM

## Zajęcia 8, 9.
Algorytm przeszukiwania z tabu (*tabu search - TS*)

### CEL ZAJĘĆ

Zapoznanie studentów z algorytmem przeszukiwania z tabu, jego implementacją dla wybranych problemów optymalizacji dyskretnej, eksperymenty obliczeniowe.

### PROBLEM 1

#### Definicja problemu

**Problem przydziału (*assignment problem*)**

Definicja jak w konspekcie do zajęć 6 i 7.

#### Algorytm przeszukiwania z tabu

**Opis**
Wykorzystamy algorytm przeszukiwania z tabu o podanym pseudokodzie.

```
algorytm tabu
ustaw listę tabu T = ∅
vc = wybierz_rozwiazanie_poczatkowe();
best = vc;
repeat
      vn = wybierz_rozwiazanie_z_otoczenia(vc);
      vc = vn;
      if (f(vn) jest lepsze od f(best))
            best = vn;
      uaktualnij_liste_tabu(T);
until (warunek_stopu)
```

… takie, które nie znajduje się na liście T

**Zmienne, kodowanie rozwiązania**
**Funkcja celu**
**Ograniczenia**
Jak w konspekcie do zajęć 6 i 7.

**Eksperyment podstawowy**
Celem eksperymentu jest znalezienie rozwiązania przybliżonego z wykorzystaniem powyższego algorytmu przeszukiwania z tabu. Eksperyment przeprowadź dla zbioru danych testowych zawartych w pliku `assign100.txt` (http://people.brunel.ac.uk/~mastjjb/jeb/orlib/assigninfo.html).

Założenia:

- Jako rozwiązanie początkowe przyjmij rozwiązanie z losowo przypisanymi zadaniami do pracowników
- Jako otoczenie bieżącego rozwiązania przyjmij zbiór rozwiązań, które można otrzymać z bieżącego rozwiązania poprzez zamianę zadań wykonywanych przez dwóch losowo wybranych pracowników. Zatem *ruch* prowadzący z bieżącego rozwiązania do pewnego rozwiązania z sąsiedztwa możemy przykładowo zdefiniować jako:

  *ruch* = (prac1, zad1, prac2, zad2)
- Jako zasadę przyjmujemy, że każdy wykonany ruch wędruje na listę tabu, czyli lista tabu zawiera ostatnio wykonane ruchy. Może zawierać max. 50 ruchów, każdy ruch przebywa na liście tabu 50 iteracji (stała kadencja) – *Uwaga! Jak reprezentować listę tabu?*
- Jako kryterium zatrzymania całego algorytmu przyjmij osiągnięcie maksymalnej liczby iteracji *MAX_ITER* = 1000.

Określ:

1. Jak wygląda przypisanie pracowników do zadań oraz jaki łączny koszt wykonania zadań przez pracowników uzyskałeś?
2. Z uwagi na niedeterministyczny charakter algorytmu, wykonaj 10 eksperymentów, a następnie wyciągnij średni koszt wykonania zadań przez pracowników.
3. Sprawdź działanie algorytmu dla następujących wartości:
   a. Długość listy tabu = 5000, kadencja = 5000
   b. Długość listy tabu = 2000, kadencja = 2000
   c. Długość listy tabu = 200, kadencja = 200
   d. Długość listy tabu = 1000, kadencja = 200
   e. Długość listy tabu = 200, kadencja = 1000

   Sprawdź poprawność (dopuszczalność) ww. wartości.

4. Sprawdź działanie algorytmu dla wartości określonych w 3a.,3 b., 3c., 3d. przy założeniu, że *MAX_ITER* = 1000, 2000, 5000.

**Modyfikacje algorytmu**

Dokonaj modyfikacji algorytmu polegających na:

1. Modyfikacji funkcji `wybierz_rozwiazanie_z_otoczenia()` tak, aby wybór rozwiązania `vn` z otoczenia `vc` następował tylko wtedy, gdy poprawia ono rozwiązanie `vc`. Wykonaj max 20 takich prób (`max_wewn_iter = 20)`.
2. Pozostawiamy ideę sąsiedztwa polegającą na wymianie zadań wykonywanych przez dwóch pracowników, ale dokonujemy modyfikacji, tak aby wybór pracowników, pomiędzy którymi chcemy dokonać wymiany zadań nie był losowy, ale opisany poprzez:
    a. Wyborze w rozwiązaniu `vc` w pierwszym kroku takiego przypisania pracownika do zadania (prac1, zad1), które daje najwyższy koszt, a następnie poszukaniu przypisania jakiegokolwiek innego pracownika do zadania (prac2, zad2) , które po zamianie wykonywanych zadań, tj. (prac1, zad2) i (prac2, zad1) spowodowałoby spadek łącznego kosztu.
    b. Możliwości uzyskania maksymalnego zysku (obniżenia kosztu) wynikającego z takiej zamiany.
    Zauważ, że opis ruchu, który wędruje na listę tabu może pozostać bez zmian.
3. *Opcjonalne:* Wprowadzeniu innego rodzaju sąsiedztwa. Jakiego?
4. *Opcjonalne:* Wprowadzeniu dodatkowego rodzaju pamięci, polegającego na zapamiętaniu dodatkowo częstości wykonywania poszczególnych ruchów. Zastanów się i nad sensownym wykorzystaniem tego rodzaju pamięci.

# PROBLEM 2

## Definicja problemu

**Problem plecakowy (*knapsack problem*) – przypomnienie z zajęć 1**
Dla danych $n$ przedmiotów, każdy o wielkości $s_i$ oraz wartości $w_i$ ($i = 1…n$), należy zadecydować, które przedmioty należy załadować do plecaka o pojemności $B$ (nie przekraczając jego pojemności), aby zmaksymalizować wartość załadowanych przedmiotów.

## Algorytm przeszukiwania z tabu z kryterium aspiracji

**Opis**
Wykorzystamy zmodyfikowany algorytm przeszukiwania z tabu o podanym pseudokodzie.

```
algorytm tabu_2
ustaw listę tabu T = ∅
vc = wybierz_rozwiazanie_poczatkowe();
best = vc;
repeat
      vn = wybierz_rozwiazanie_z_otoczenia(vc);
      vc = vn;
      if (f(vn) jest lepsze od f(best))
            best = vn;
      uaktualnij_liste_tabu(T);
until (warunek_stopu)
```

… takie, które nie znajduje się na liście T lub spełnia kryterium aspiracji

**Zmienne, kodowanie rozwiązania – przypomnienie z zajęć 1**
Wektor zmiennych $x = (x_1, x_2, ..., x_n)$, $x_i \in \{0, 1\}$, gdzie 1 oznacza, że przedmiot umieszczamy w plecaku, a 0 – nie umieszczamy.

**Funkcja celu – przypomnienie z zajęć 1**
Suma wartości przedmiotów zapakowanych do plecaka (max)

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

**Ograniczenia – przypomnienie z zajęć 1**
Nie przekraczamy pojemności plecaka

$$\sum_{i=1}^{n} s_i x_i \leq b$$

**Eksperyment podstawowy**
Celem eksperymentu jest znalezienie rozwiązania przybliżonego z wykorzystaniem powyższego algorytmu przeszukiwania z tabu z kryterium aspiracji. Eksperyment przeprowadź dla zbioru danych testowych zawartych w pliku `f1_l-d_kp_10_269` (http://artemisa.unicauca.edu.co/~johnyortega/instances_01_KP/).

Format pliku z danymi:

*Nazwa pliku :*
`kp_n_B`, gdzie kp: nazwa instancji, n: liczba przedmiotów, B: pojemność plecaka

*Format pliku:*
n B
w1 s1
w2 s2
…
wn sn

gdzie: wi: wartość i-tego przedmiotu, si: wielkość i-tego przedmiotu

Założenia:

- Jako rozwiązanie początkowe (funkcja `wybierz_rozwiazanie_poczatkowe()`) przyjmij rozwiązanie z losowo załadowanymi przedmiotami do plecaka, przy założeniu nieprzekroczenia pojemności plecaka.
- Jako otoczenie bieżącego rozwiązania (wykorzystywane m.in. w funkcji `wybierz_rozwiazanie_z_otoczenia()`) przyjmij zbiór rozwiązań, które można otrzymać z bieżącego rozwiązania poprzez usunięcie z plecaka dowolnego losowo wybranego przedmiotu (ozn. *przedmiot_1*) z jednoczesnym załadowaniem do plecaka innego przedmiotu spośród dotychczas nie załadowanych (ozn. *przedmiot_2*), pod warunkiem, iż rozwiązanie to poprawia wartość funkcji celu aktualnego rozwiązania oraz nie zostanie przekraczana pojemność plecaka. Zatem *ruch* prowadzący z bieżącego rozwiązania do pewnego rozwiązania z sąsiedztwa możemy zdefiniować jako:

  $$ruch = (przedmiot\_1, przedmiot\_2)$$

- Jako zasadę przyjmujemy, że każdy wykonany ruch wędruje na listę tabu, czyli lista tabu zawiera ostatnio wykonane ruchy, a rozwiązanie, które może być wybrane z otoczenia (`wybierz_rozwiazanie_z_otoczenia()`) dotyczy tylko rozwiązań/ruchów nie znajdujących się na liście tabu oraz **dodatkowo zakładamy, iż przy wybieraniu rozwiązania z otoczenia możliwe jest wybranie rozwiązania, które znajduje się na liście tabu, gdyby poprawiało ono globalnie najlepsze znalezione do tej pory rozwiązanie (kryterium aspiracji).**
- Lista tabu może zawierać max. 10 ruchów, każdy ruch przebywa na liście tabu 10 iteracji (stała kadencja).
- Jako kryterium zatrzymania całego algorytmu przyjmij osiągnięcie maksymalnej liczby iteracji *MAX_ITER* = 10000.

Znajdź:

a) Najlepsze rozwiązanie ze wskazaniem wartość funkcji celu tego rozwiązania
b) Jakość tego rozwiązania mierzoną średnim błędem względnym MRE (*mean relative error*), czyli o ile procent jest ono gorsze od rozwiązania optymalnego? (Rozwiązanie optymalne -> patrz poniżej Dodatek)
c) Dobierz wartości parametrów: długość listy tabu, długość kadencji oraz maksymalną liczbę iteracji dającą jak najlepsze rezultaty.

Przeprowadź rozszerzony eksperyment, który pozwoliłby dobrać jak najlepszy sposób tworzenia rozwiązania początkowego oraz wyboru rozwiązania z otoczenia, rozważając następujące warianty:

| Warianty | `wybierz_rozwiazanie_poczatkowe()` | `wybierz_rozwiazanie_z_otoczenia()` |
|---|---|---|
| Wariant 1 | Tworzę rozwiązanie poprzez wybór losowy przedmiotów do załadowania do plecaka, przy założeniu nieprzekroczenia pojemności plecaka (jak wyżej) | Wybieram rozwiązanie, które można otrzymać z bieżącego rozwiązania poprzez usunięcie z plecaka dowolnego losowo wybranego przedmiotu z jednoczesnym załadowaniem do plecaka innego przedmiotu spośród dotychczas nie załadowanych, pod warunkiem, iż rozwiązanie to poprawia wartość funkcji celu aktualnego rozwiązania oraz nie zostanie przekraczana pojemność plecaka (jak wyżej) |
| Wariant 2 | Tworzę rozwiązanie poprzez wybór po kolej przedmiotów do załadowania, dla których iloraz $w_i/s_i$ będzie największy, przy założeniu nieprzekroczenia pojemności plecaka | Wybieram rozwiązanie, które można otrzymać z bieżącego rozwiązania poprzez usunięcie z plecaka dowolnego losowo wybranego przedmiotu z jednoczesnym załadowaniem do plecaka innego przedmiotu spośród dotychczas nie załadowanych, pod warunkiem, iż rozwiązanie to poprawia wartość funkcji celu aktualnego rozwiązania oraz nie zostanie przekraczana pojemność plecaka |
| Wariant 3 | Tworzę rozwiązanie poprzez wybór losowy przedmiotów do załadowania do plecaka, przy założeniu nieprzekroczenia pojemności plecaka | Zaproponuj inne sensowne sąsiedztwo |
| Wariant 4 | Tworzę rozwiązanie poprzez wybór po kolej przedmiotów do załadowania, dla których iloraz $w_i/s_i$ będzie największy, przy założeniu nieprzekroczenia pojemności plecaka | Zaproponuj inne sensowne sąsiedztwo |

Eksperyment przeprowadź na dołączonych danych testowych **knapsack_instances.zip** (http://artemisa.unicauca.edu.co/~johnyortega/instances_01_KP/), a wyniki przedstaw w sprawozdaniu w dwóch plikach (osobno dla małych i dużych instancji): **Raport_Konspekt_08_09_small_instances.xlsx** oraz **Raport_Konspekt_08_09_large_instances.xlsx**

**Dodatek. Rozwiązania optymalne:**

Instancje o małym rozmiarze

| Plik | Optimum |
|---|---|
| f1_l-d_kp_10_269 | 295 |
| f2_l-d_kp_20_878 | 1024 |
| f3_l-d_kp_4_20 | 35 |
| f4_l-d_kp_4_11 | 23 |
| f5_l-d_kp_15_375 | 481,0694 |
| f6_l-d_kp_10_60 | 52 |
| f7_l-d_kp_7_50 | 107 |
| f8_l-d_kp_23_10000 | 9767 |
| f9_l-d_kp_5_80 | 130 |
| f10_l-d_kp_20_879 | 1025 |

Instancje o dużym rozmiarze

| Plik | Optimum |
|---|---|

| | |
|---|---|
| knapPI_1_100_1000_1 | 9147 |
| knapPI_1_200_1000_1 | 11238 |
| knapPI_1_500_1000_1 | 28857 |
| knapPI_1_1000_1000_1 | 54503 |
| knapPI_1_2000_1000_1 | 110625 |
| knapPI_1_5000_1000_1 | 276457 |
| knapPI_1_10000_1000_1 | 563647 |