

Instytut Informatyki Stosowanej  
Politechnika Łódzka

## **PODSTAWY SZTUCZNEJ INTELIGENCJI**

**Laboratorium**

### **ROZWIĄZYWANIE PROBLEMÓW POPRAZ PRZESZUKIWANIE PRZESTRZENI STANÓW**

Opracowanie:  
Dr hab. inż. Jacek Kucharski  
Dr inż. Piotr Urbanek

## Cel ćwiczenia

Przedmiotem ćwiczenia jest analiza efektywności różnych metod rozwiązywanie problemu poprzez przeszukiwanie przestrzeni rozwiązań (stanów). Analiza ta realizowana ma być na przykładzie problemu komiwojażera.

## Zakres ćwiczenia

1. W środowisku MATLAB należy napisać zestaw  $m$ -plików umożliwiających modelowanie i rozwiązanie problemu komiwojażera dla dowolnej zadanej liczby  $N$  miast położonych w różnych punktach na płaszczyźnie  $xy$ .
  - a.  $m$ -plik skryptowy (np. `szukaj.m`) pełniący rolę programu głównego, w którym należy uwzględnić następujące elementy:
    - i. zapis parametrów poszczególnych miast w postaci trójek ( $nr\_miasta$ ,  $współrzędna\_x$ ,  $współrzędna\_y$ ) (np. w macierzy `miasta` o wymiarach  $N \times 3$ );
    - ii. iteracyjne sterownie przeszukiwaniem umożliwiające przegląd całej przestrzeni stanów  $S$ , tj. wszystkich kombinacji ścieżek (np. w formie pętli `while`);
    - iii. reprezentację stanu  $s$  przestrzeni przeszukiwania w formie zmiennej strukturalnej, w której polach zapisane są informacje o dotychczasowym przebiegu procesu szukania i ocena jakości danego węzła, tj. ścieżka prowadząca do danego miasta (np. w formie wektora wierszowego `ścieżka`), jej koszt (długość) oraz ocena heurystyczna ( pola skalarne `koszt` i `heurystyka`)
    - iv. zapamiętywanie (np. w formie dwóch tablic komórkowych) zbioru węzłów pozostałych w danej chwili do sprawdzenia oraz zbioru węzłów już sprawdzonych, z możliwością ich porządkowania;
    - v. graficzną ilustrację procesu szukania poprzez wizualizację ścieżki, rozmiaru wykorzystywanej pamięci (długości poszczególnych list) itd.
  - b.  $m$ -plik funkcyjny ( np. `potomstwo.m`) pozwalający wygenerować kolejne stany (potomne) bieżącego stanu poprzez połączenie bieżącego miasta ze wszystkimi nie odwiedzionymi w dotychczasowym procesie szukania miastami. Funkcja powinna:
    - i. mieć dwa argumenty wejściowe (węzeł bieżący i tablice zawierającą wszystkie miasta) oraz jeden argument wyjściowy (listę wygenerowanych węzłów potomnych)
    - ii. sprawdzać, które miasta nie znajdują się na ścieżce reprezentującej bieżący stan i tworzyć listę tych miast (np. w formie wektora wierszowego)
    - iii. generować listę bezpośrednich węzłów potomnych bieżącego węzła poprzez dodawanie do aktualnej ścieżki kolejnych nie odwiedzionych miast
    - iv. obliczać koszt każdego nowotworzonego węzła w postaci sumarycznej długości ścieżki w nim zawartej tzn:

$$koszt = \sum_{i=2}^M \sqrt{(x_{i-1} - x_i)^2 + (y_{i-1} - y_i)^2}$$

gdzie  $M$  - jest liczbą miast zapisanych w ścieżce danego węzła

Uwaga: należy uwzględnić fakt, że koszt nowego stanu powstaje w wyniku zwiększenia kosztu stanu bieżącego o odległość do dodanego w danym stanie miasta

- v. obliczać wartość oceny heurystycznej każdego nowotworzonego węzła w postaci zależnej od przyjętej strategii przeszukiwania.
- c. m-plik funkcyjny (np. sortuj.m) umożliwiający uporządkowanie węzłów na liście według określonego kryterium. Funkcja ta powinna:
  - i. posiadać dwa argumenty wejściowe (lista węzłów do sortowania i kryterium) oraz jeden argument wyjściowy (lista posortowanych węzłów)
  - ii. implementować prosty algorytm sortujący (np. algorytm bąbelkowy)
2. Wprowadzić do programu elementy charakterystyczne dla szukania systematycznego (odpowiednie zarządzanie listą węzłów do odwiedzenia) i przeanalizować przebieg procesu szukania metodami w głąb i wszerz
3. Wykorzystując funkcję sortującą przeanalizować przebieg procesu szukania metodą najmniejszego kosztu
4. Wprowadzić do funkcji generującej potomstwo różne sposoby obliczania funkcji heurystycznej i przeanalizować przebieg procesu szukania metodami heurystycznymi: zachłanną i A\*