Piotr Małek, nr 199322

Filip Piechocki, nr 199334

**Projekt z przedmiotu GIS**

**„Problem komiwojażera”**

**Sprawozdanie nr. 1**

# Opis problemu:

Problem komiwojażera (ang. Traveling Salesman Problem, w skrócie TSP) polega na wyznaczeniu najkrótszej ścieżki przebiegającej przez wszystkie wierzchołki w grafie, czyli znalezieniu minimalnego cyklu Hammiltona w grafie spójnym. Nazwa nawiązuje do najbardziej nasuwającego się zastosowania algorytmów wyznaczających taką ścieżkę, czyli wyliczania najkrótszej trasy dla samochodu dostawczego między lokalizacjami do których musi dostarczyć towar, tak aby zużyć jak najmniej benzyny.

Mimo że problem jest powszechny i wykracza daleko poza zagadnienia logistyczne, nie wynaleziono jeszcze optymalnego algorytmu działającego w rozsądnym czasie i wyznaczającego najlepszą możliwą ścieżkę. Wymyślono za to rozmaitość rozwiązań heurystycznych, działąjących w akceptowalnie długim czasie i dających bardzo dobre wyniki, choć bez gwarancji że nie ma rozwiązań lepszych.

Problem komiwojażera ma sens dla grafu spójnego, skierowanego bądź nie skierowanego, ważonego.

Jako przykład rozwiązania heurystycznego zostanie w ramach projektu zaimplementowany algorytm kopca mrówek, oraz naiwny do celów porównawczych.

# Założenia projektowe:

W ramach projektu zostaną zaimplementowane dwie metody wyznaczania minimalnego cyklu Hammiltona w grafie: naiwy oraz heurystyczny wykorzystujący system kopca mrówek.

Projekt zostanie zrealizowany w C++, z wykorzystaniem biblioteki Qt do celów prezentacji wyników.

Program który będzie wynikiem tego projektu będzie umożliwiał zaczytanie grafu z pliku tekstowego oraz zapewni mechanizm do prezentacji wyników. Program zostanie napisany z użyciem mechanizmu pluginów, który umożliwi jego łatwą rozbudowę o pluginy implementujące kolejne algorytmy grafowe. Efektem tego projektu będzie aplikacja i dwa pluginy implementujące algorytm heurystyczny z wykorzystanie systemu kopca mrówek i bruteforce.

# Opis algorytmu kopca mrówek:

Rozwiązanie zostanie zaimplementowane zgodnie ze specyfikacją zawartą w publikacji *„Ant Colony System: A Cooperative Learning Approach to the Traveling Salesman Problem”* [[1]](#footnote-1) .

## Ogólna koncepcja:

Koncepcja tego rozwiązania sprowadza się do zaimplementowania rozwiązania bazującego na tym jakie wykorzystują w naturze mrówki. Chodzi o to aby graf zasiedlić populacją botów (mrówek). Każda z mrówek zostawia na swojej drodze feromony, a każda kolejna mrówka przy podjęciu decyzji którą drogą pójść decyduje się na nią zgodnie z prawdopodobieństwem bazującym na ilości pozostawionych na niej feromonów. Czyli teraz jeżeli rozpatrzymy dwie ścieżki między dwoma wierzchołkami, jedną o długości 10 a drugą 20 to w pierwszej fazie działania algorytmu na żadnej nie będzie feromonów dlatego połowa mrówek pójdzie ścieżką krótszą, a połowa dłuższą. Jednak ponieważ mrówki poruszają się z pewną stałą prędkością to szybciej przejdą przez krótszą ścieżkę a co za tym idzie zostawią na niej szybciej feromony. Dlatego kolejne mrówki przy podjęciu decyzji którą ścieżką pójść, chętniej wybiorą krótszą trasę, ponieważ szybciej na niej będą gromadzić się feromony, a co za tym idzie potem druga, dłuższa ścieżka przestanie być przez mrówki faktycznie wybierana. Po zakończeniu przez mrówki swoich cykli następuje aktualizacja ilości feromonów. Proporcjonalnie część feromonów wyparowuje z krawędzi, a następnie każda mrówka zostawia dodatkowe feromony (ilość w zależności od długości trasy) na swojej ścieżce, tak że krótsze trasy są promowane większą ilością feromonów. Proces się powtarza. Po kilku iteracjach mrówka której cykl jest najkrótszy, jest posiadaczem optymalnej ścieżki i rozwiązaniem prolemu. Formalny i dokładny opis działania algorytmu znajduje się w publikacji na której bazuje pomysł rozwiązania.

## Założenia projektowe:

Inicjalny graf będzie pobierany z pliku w formacie xml o następującej strukturze:

<graph>

<edge>

<vertex>A</vertex>

<vertex>B</vertex>

<weight>3</weight>

</edge>

<edge>

<vertex>C</vertex>

<vertex>B</vertex>

<weight>8</weight>

</edge>

<edge>

<vertex>A</vertex>

<vertex>C</vertex>

<weight>12</weight>

</edge>

</graph>

Gdzie tag graph przechowuje informacje o całym grafie, czyli :

1. edge – informacja o krawędziach:
   1. vertex – etykieta wierzchołka pierwszego
   2. vertex – etykieta wierzchołka drugiego
   3. weight – waga krawędzi (odległość pomiędzy wierzchołkami)

Wynik przetwarzania zostanie przedstawiony w programie jako porównanie ścieżki najlepszej wyznaczonej metodą bruteforce oraz efektu przetwarzania z wykorzystaniem wirtualnego kopca mrówek.

Zakładamy że graf na którym będziemy pracować będzie grafem spójnym. Dlatego przed poddaniem grafu działaniu algorytmu zostanie przetestowany pod kątem spójności przy użyciu zasady DFS (wybieramy losowy wierzchołek i przechodzimy graf w głąb).

Ponieważ problem komiwojażera zakłada odwiedzanie każdego z miast tylko raz, to spójność grafu nie jest jedynym warunkiem na istnienie cyklu Hamiltona. Np. z wierzchołka będącego liściem nie da się już wrócić, wiec przed poddaniem grafu działaniu algorytmu, zostanie on zamieniony w graf pełny, gdzie każda dodana podczas tej operacji krawędź grafu pomiędzy dwoma wierzchołkami będzie miała wagę równą wadze najkrótszej ścieżki z znalezionej pomiędzy tymi wierzchołkami.

## Struktury danych:

W celu jak najlepszej optymalizacji graf będzie jako tablica wierzchołków. Wierzchołek będzie przechowywał informację o wychodzących z niego krawędziach (słownik, o kluczu reprezentowanym wierzchołek docelowy). Każda krawędź z kolei będzie przechowywać informacje o swojej długości, oraz o poziomie feromonów w niej zawartych.

Wirtualne mrówki będą przetrzymywane w dwóch liście i będą przechowywały informacje o odwiedzonych wierzchołkach oraz długość ścieżki.

Na etapie implementacji niektóre z założeń mogą ulec zmianie.

1. Publikacja dostępna pod adresem http://www.idsia.ch/~luca/acs-ec97.pdf [↑](#footnote-ref-1)