

Data oddania sprawozdania: 16.01.2018

Termin zajęć: WT 730- 900

Sala: L2.6 (C-16)

**SPRAWOZDANIE**

**Projektowanie Efektywnych Algorytmów**

**Prowadzący:** Dr inż. Łukasz Jeleń

**Autor:** Paweł Szynal 226026

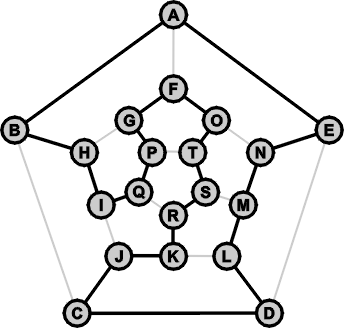
226026@student.pwr.edu.pl

### Wrocław 2017

# Wstęp

Celem projektu była implementacja i analiza efektywności algorytmu Tabu Search dla problemu komiwojażera (TSP).

# Cykl Hamiltona

Jest to śceiżka w grafie która odwiedza każdy jego wierzchołek dokładnie raz. Problem ten zalicza się to tzw. problemów NP zupełnych. Dla dużej liczby wierzchołków jest on praktycznie nierozwiązywalny w sensownym czasie.

Rysunek 1 Cykl Hamiltona:

A B H I Q P G F O T S R K J C D L M N E A

# Problem Komiwojażera

Problem komiwojażera przedstawiony jest następująco: dana jest liczba miast. Komiwojażer musi odwiedzić każde miasto dokładnie jeden raz, a następnie wrócić do punktu początkowego. Znany jest koszt przejazdu między każdą parą miast. Należy zaplanować drogę komiwojażera tak, aby każde miasto zostało odwiedzone dokładnie jeden raz oraz by całkowity koszt przejazdu był jak najmniejszy. Problem ten można przestawić za pomocą grafu pełnego, tzn. grafu o stuprocentowym nasyceniu krawędziowym, co oznacza, że każda para wierzchołków jest połączona krawędzią. Miasta, które musi odwiedzić komiwojażer są wierzchołkami, a drogi łączące te miasta to krawędzie z wagami, symbolizującymi koszt podróży daną drogą. Rozwiązanie tego problemu istnieje zawsze, ponieważ dowolny graf pełny posiada co najmniej jeden cykl Hamiltona. Ponieważ graf ma skończona liczbę wierzchołków, to w zbiorze cykli Hamiltona istnieje taki (niekoniecznie jedyny), który posiada minimalna sumę wag krawędzi.



## Symetryczny problem komiwojażera

Dla dowolnych miast A i B odległość z A do B jest taka sama jak z B do A.



## Asymetryczny problem komiwojażera

O między miastami A i B może być różna niż odległość z miasta B do miasta A.

# Rozwiązanie problemu komiwojażera

**Tabu Search - opis działania algorytmu**

Algorytm 'tabu search' (TS) to metoda przeszukiwania zbioru rozwiązań naśladująca proces wykonywany przez człowieka. Główna koncepcja jest bardzo prosta. „Pamięć” zmusza algorytm poszukiwania do badania nowych obszarów przestrzeni przeszukiwania. Za każdym razem wybieramy najlepsze lokalne rozwiązanie.

Możemy zapamiętać kilka z analizowanych ostatnio rozwiązań lub rozwiązania bezpośrednio je przypominające – stają się ona tak zwanymi punktami tabu, które omijamy. Przeszukiwanie 'tabu search' jest więc w gruncie rzeczy deterministyczne, ale można je wzbogacić o elementy probabilistyczne. Algorytm rozpoczyna się od wylosowania początkowej listy rozwiązań- przyjmuje się to jako pierwsze: (do tej pory najlepsze) rozwiązanie, następnie poszukiwane jest lepsze rozwiązanie wykorzystując do tego metody przeszukiwania lokalnego (w tym przypadku: metody sąsiedztwa).

Sąsiedztwem (otoczeniem) rozwiązania ∏ nazywamy zbiór permutacji otrzymanych z ∏ poprzez wykonanie zaburzeń swap. Rozwiązanie sąsiednie jest wybierane z sąsiedztwa rozwiązania xi w taki sposób, aby ominąć lub opuścić minimum lokalne.

W kolejnych iteracjach i z sąsiedztwa aktualnego rozwiązania T(Xi) wybieramy „niezakazane” rozwiązanie xi+1 takie, że: Droga(xi+1) = min {x є T(xi)} (Droga(x)). Wybrane rozwiązanie jest zapamiętywane (najczęściej na liście tabu) i w dalszym procesie poszukiwań posiada status 'tabu'. W pamięci utrzymujemy liczbę iteracji, przez które dana zmiana pozostanie nadal na liście tabu.

Lista tabu jest (w tym przypadku) w rzeczywistości tablicą, która przechowuje informacje odnośnie „zakazanych” ruchów. Zakaz ten wynika z faktu, iż ponowne wykonanie tej samej operacji jest w większości przypadków bezcelowe, gdyż nie osiągnie bardziej optymalnego rozwiązania. Ruchy określane jako zakazane definiuje się jako takowe po tym, gdy rozwiązanie znalezione w wyniku wykonania jednego z trzech możliwych rodzajów zaburzeń jest lepsze od dotychczasowego najlepszego rozwiązania.

Zaburzenia te wykonywane są w następujący sposób: Zaburzenie typu swap(i,j)- polega na zamianie miejscami miast i oraz j

Przykład: Droga 0 1 2 3 4 0 przy zaburzeniu swap (1,3) będzie porównywać optymalność rozwiązania 0 3 2 1 4

Droga 0 1 2 3 4 0 przy zaburzeniu insert (0,4) będzie wyglądała następująco:

0 3 2 1 4 0

# Pomiary

**Dane i wykresy dla pliku br17.atsp**

**Optymalne rozwiązanie: 39**

|  |  |  |
| --- | --- | --- |
| **Koszt** | **Błąd względny** | **Czas[ns]** |
| 68 | 74,36% | 7818887 |
| 58 | 48,72% | 11158182 |
| 53 | 35,90% | 12294887 |

**Dane i wykresy dla pliku ft70.atsp**

**Optymalne rozwiązanie: 38673**

|  |  |  |
| --- | --- | --- |
| **Koszt** | **Błąd względny** | **Czas[ns]** |
| 48153 | 24,51% | 3429102 |
| 46374 | 19,91% | 8238159 |

**Dane i wykresy dla pliku eil51.tsp**

**Kosz optymalnej drogi:426**

|  |  |  |
| --- | --- | --- |
| **Koszt** | **Błąd względny** | **Czas[ns]** |
| 735 | 72,54% | 2530002 |
| 664 | 55,87% | 34483466 |
| 661 | 55,16% | 37140738 |

# Wnioski

Jak można zauważyć na podstawie tabel i wykresów, algorytm tabu search

(o ile nie otrzyma odpowiedniej ilości czasu) nie jest algorytmem dającym najbardziej optymalny wynik, a czas wymagany do uzyskania dobrego wyniku jest tym większy, im więcej miast należy rozpatrzyć.