Adrian Frydmański 209865 Data oddania: **21 XII 2015r.**Dawid Gracek 209929 Prowadzący: **dr inż. Tomasz Kapłon**

Zadanie projektowe 2

Projektowanie efektywnych algorytmów

**Temat: „Algorytm tabu search dla problemu komiwojażera”**

# Zadanie

Należy zaimplementować algorytm tabu search dla problemu komiwojażera oraz dokonać testów polegających na pomiarze czasu działania algorytmu w zależności od wielkości instancji oraz jakości dostarczanych rozwiązań (należy porównać rozwiązania dostarczone przez algorytm z najlepszymi znanymi rozwiązaniami dla przykładów testowych i z wynikami z poprzedniego zadania).

# Informacje wstępne

Problem komiwojażera opisuje znajdywanie minimalnego cyklu Hamiltona w grafie pełnym. Innymi słowy: komiwojażer ma za zadanie obejść wszystkie miasta, mając do dyspozycji odległości między nimi. Musi to zrobić jak najkrótszą drogą.

Każdy wierzchołek grafu jest miastem, przez które musi przejechać komiwojażer. Grupę *n* miast reprezentuje zbiór *N = {1, 2, …, n}*. Drogi pomiędzy miastami reprezentowane są przez macierz   
*D = {dij, i є N, j є N, i ≠ j}*, gdzie *dij* jest odległością pomiędzy miastem *i* oraz *j*.

Są dwie odmiany problemu – symetryczna i asymetryczna. W symetrycznej droga z miasta a do b jest równa drodze z miasta b do a. W asymetrycznej drogi te mogą być różne. W macierzy odległości między miastami wygląda to następująco:

* dla problemu symetrycznego *dij = dji*,
* dla problemu asymetrycznego *dij ≠ dji*.

Algorytm tabu search związany jest z pojęciem sąsiedztwa punktu. Dla n-wymiarowej przestrzeni jest to hipersfera o infinitezymalnym promieniu *ε* i środku w punkcie *x*. W n-elementowej permutacji wyróżnia się 3 typy sąsiedztw:

* insert – element „wyjęty” ze swojego miejsca i „wsadzony” pomiędzy dwa inne elementy,
* swap – dwa elementy zamienione ze sobą,
* invert – ciąg kolejnych k elementów poukładany w drugą stronę.

Autorem algorytmu jest profesor Fred Gloger. Jego sposób bazuje na losowym przeszukiwaniu, jednakże dodatkowym elementem jest tu tzw. „lista tabu”, dzięki której unika się utknięcia w minimum lokalnym.

Algorytm tabu search rozpoczyna działanie od losowego rozwiązania początkowego x0 i porusza się po sąsiednich rozwiązaniach. W danej iteracji z sąsiedztwa aktualnego rozwiązania *N(xi)* wybierane jest „niezakazane” rozwiązanie *x(i+1)* takie, że: *f(x(i+1)) = min {f(x)}*, gdzie *xϵN(xi)*, a *f(x)* jest funkcja celu. Rozwiązanie odwiedzone, zostaje zapamiętane i – mając status tabu – w dalszym procesie poszukiwania jest zakazane.

Zaimplementowany algorytm przewiduje długość listy tabu równą wielkości instancji ze względu na użyte ograniczenie, jakim jest wprowadzanie do listy ostatniego poprawnego wygenerowanego wierzchołka. Lista tabu jest kolejką FIFO. Zbytnie wydłużenie listy tabu może skutkować utknięciem i przeszukiwaniem w obrębie jednego lokalnego minimum. Metodą dywersyfikacji jest losowy wybór wierzchołków. Można było użyć metody polegającej na wykorzystaniu pamięci długoterminowej odpowiadającej za przechowywanie licznika „odwiedzeń” danej pary wierzchołków. Na jej podstawie, po zadanej liczbie iteracji, można wygenerować nowe rozwiązanie startowe złożone z par najrzadziej odwiedzonych. Współczynnik odpowiadający za liczbę permutacji – tau – ma ścisły wpływ na efektywność algorytmu. Jeżeli jest za mały, te same rozwiązania sprawdzane są więcej, niż raz. Jeżeli zaś za duży, spada wydajność algorytmu.

# Implementacja i Testowanie

Program został napisany w języku C++ za pomocą środowiska Visual Studio 2015. Program realizuje algorytm symulowanego wyżarzania dla problemu komiwojażera.

Algorytm dla każdego pliku wejściowego był wykonywany 10 razy, a następnie jego wyniki zostały uśrednione. Testy były przeprowadzane dla 4, 5, 10, 14, 17, 26, 29, 42, 48 i 76 miast (pliki załączone wraz z programem). Pliki wejściowe mogą być w dwóch postaciach:

* sposób dra Mierzwy: liczba miast, a następnie kolejne wartości z macierzy odległości między nimi;
* plik TSP ze współrzędnymi miast – liczba miast podana po „DIMENSION:” (lub „DIMENSION :”), współrzędne i numery wierzchołków podawane trójkami po „NODE\_COORD\_SECTION”.

# Czas wykonywania się algorytmu był mierzony dzięki własnej klasie MyTimer. Zlicza ona takty zegara procesora i dzieli tę liczbę przez jego częstotliwość, a następnie mnoży przez 1000 aby wynik był podany w ms.

# wyniki

Współczynnik odpowiadający za liczbę permutacji był dobierany automatycznie na podstawie rozmiaru instancji i był równy jej kwadratowi pomnożonemu przez 100.

Poniżej przedstawiono wyniki pomiarów:

Tabela 1 długości wyznaczonych tras dla obu algorytmów

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Liczba miast | Symulowane wyżarzanie | | | | Tabu search | Optimum | Wzrost t.s. względem opt. | Spadek t.s. względem s.w.(0,999) |
| 0,9 | 0,95 | 0,995 | 0,999 |
| 4 | 200 | 200 | 200 | 200 | 300 | 200 | 150% | -33% |
| 5 | 22 | 21 | 17 | 16 | 16 | 16 | 100% | 0% |
| 10 | 331 | 340 | 281 | 270 | 212 | 212 | 100% | 27% |
| 14 | 49 | 46 | 45 | 40 | 39 | 33 | 118% | 3% |
| 17 | 3824 | 3585 | 3303 | 3285 | 2869 | 2085 | 138% | 14% |
| 26 | 2306 | 2195 | 1948 | 1904 | 1140 | 937 | 122% | 67% |
| 29 | 5040 | 4977 | 4647 | 4354 | 4236 | 2020 | 210% | 3% |
| 42 | 4159 | 4159 | 3869 | 3719 | 1699 | 1273 | 55% | 432% |
| 48 | 135279 | 127276 | 121034 | 116964 | 108310 | 10628 | 1019% | 8% |
| 76 | 171510 | 179728 | 173609 | 179123 | 150780 | 108159 | 139% | 19% |

Tabela 2 czas działania algorytmów [ms]

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Liczba miast | Symulowane wyżarzanie | | | | Tabu search |
| 0,9 | 0,95 | 0,995 | 0,999 |
| 4 | 0 | 1 | 17 | 90 | 2 |
| 5 | 1 | 2 | 28 | 144 | 5 |
| 10 | 4 | 9 | 97 | 486 | 19 |
| 14 | 7 | 17 | 179 | 825 | 41 |
| 17 | 9 | 22 | 231 | 1055 | 74 |
| 26 | 19 | 40 | 408 | 2194 | 270 |
| 29 | 24 | 50 | 512 | 2627 | 320 |
| 42 | 37 | 78 | 792 | 4007 | 1138 |
| 48 | 50 | 104 | 1063 | 5289 | 1868 |
| 76 | 98 | 202 | 2070 | 10363 | 7677 |

Wykres 1 Zależność czasu wykonywania algorytmu tabu search od liczby miast

Wykres 2 Porównanie czasu wykonywania algorytmu tabu search i symulowanego wyżarzania (dla czterech wartości współczynnika chłodzenia)

Wykres 3 Porównanie znalezionych rozwiązań z optimum (przeglądem zupełnym)

Wykres 4 Porównanie błędów względem rozwiązania optymalnego dla obu algorytmów

Jak widać, w grafie 48-wierzchołkowym błąd był dość spory, co może wynikać z natrafienia na pesymistyczny przypadek.

# Podsumowanie i wnioski

Przygotowanie projektu pozwoliło zapoznać się z tematyką algorytmiki metaheurystycznej w postaci innej niż w przypadku symulowanego wyżarzania.

Z otrzymanych danych widać spodziewany wzrost czasu oczekiwania na wyniki. Tabu search dla większych instancji jest szybszy niż wyżarzanie z bardzo wysokim współczynnikiem chłodzenia (0,999), co widać w tabeli 2. Również jakość otrzymanych wyników jest lepsza (tabela 1).

Podobnie jak w przypadku SA pojawiają się więc dwa główne problemy – problem złożoności czasowej oraz problem jakości otrzymywanych rozwiązań. Jeśli tracimy zwiększając czas działania algorytmu zyskujemy na jakości rozwiązań oraz odwrotnie, gdy tracimy na jakości zyskujemy na czasie.

Zarówno symulowane wyżarzanie jak i tabu search opracowano aby uniknąć lokalnych minimów. Różnią się one jednak sposobem osiągania tego celu. Poszukiwanie z tabu z reguły wykonuje ruchy do góry tylko wówczas, gdy utknie w minimum lokalnym, podczas gdy symulowane wyżarzanie może 'wspinać się'  
w dowolnym momencie. Ponadto symulowane wyżarzanie jest algorytmem stochastycznym, TS – deterministycznym.

# Bibliografia

* Grzegorz Studziński, „Heurystyka, co to jest? Praktyczne przedstawienie algorytmu poszukiwania Tabu Search”, http://marek.piasecki.staff.iiar.pwr.wroc.pl/dydaktyka/isa/2007/Studzinski\_Grzegorz.pdf