Adrian Frydmański 209865 Data oddania: **24 XI 2015r.**Dawid Gracek 209929 Prowadzący: **dr inż. Tomasz Kapłon**

Zadanie projektowe 1:

Projektowanie efektywnych algorytmów

**Temat: „Algorytm symulowanego wyżarzania dla problemu komiwojażera”**

# Zadanie

Należy zaimplementować algorytm symulowanego wyżarzania 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). Wskazane jest, aby algorytm był opracowany w ten sposób, aby jego parametry były automatycznie dobierane w zależności od rozwiązywanej instancji.

# 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 i 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 symulowanego wyżarzania bazuje na analogii do tegoż procesu w rzeczywistości. Prowadzi on do uzyskania wyniku bliskiego (lub równego) optimum. Nawiązuje on do zastygania cieczy tworzącej strukturę krystaliczną. W wysokiej temperaturze cząsteczki poruszają się swobodnie, a wraz z jej spadkiem robią to coraz wolniej. Operacja powolnego schładzania ma doprowadzić metal do równowagi termodynamicznej.

Algorytm polega na generowaniu kolejnych rozwiązań i porównywania z najlepszym dotychczasowym. Rozpoczynamy od początkowego rozwiązania, losowego, a następnie poprzez modyfikacje kolejności pojedynczych miast, przechodzimy do rozwiązań sąsiednich.

Sąsiedztwem danego rozwiązania nazywamy zbiór permutacji otrzymanych z niego poprzez zaburzenie go (na przykład przez zamianę dwóch elementów – i i j). Wybór rozwiązania sąsiedniego jest dokonywany tak, by ominąć, lub opuścić minimum lokalne.

Temperatura reguluje akceptację nowych rozwiązań i maleje według jednego z poniższych schematów studzenia. W projekcie przyjęto schemat geometryczny: Ti+1 = αiTi, 0 < αi < 1.

Z każdą iteracją zachodzi generacja i ewentualna akceptacja nowego rozwiązania.

Powyższy algorytm można przedstawić następująco:

Pseudokod dla algorytmu symulowanego wyżarzania [1]

wybierz losowo rozwiązanie A

while(T > Tmin)

{

stwórz rozwiązanie B przez zamianę dwóch losowych elementów w A

if (Kryterium(B) < Kryterium(A))

A = B

else if (random(0,1) < P())

A = B

T = G()

}

Widoczna powyżej funkcja P() oznacza prawdopodobieństwo. Liczymy jej wartość z następującego wzoru:

Funkcja G() z każdą iteracją „chłodzi” graf zgodnie ze wzorem:

# 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. Temperatura początkowa wynosiła 1000, końcowa 0,001, a współczynnik chłodzenia odpowiednio: 0,9, 0,95, 0,0995 i 0,999. 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 przez wywoływanie funkcji QueryPerformanceCounter(&start) i QueryPerformanceCounter(&stop) i odjęcie od siebie start i stop dla uzyskania różnicy czasu. Wynik mnożony był przez 1000 dla uzyskania czasu w milisekundach.

# wyniki

Poniżej przedstawiono wyniki pomiarów:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Liczba miast | Współczynnik chłodzenia | | | | Przegląd zupełny |
| 0,9 | 0,95 | 0,995 | 0,999 |
| 4 | 200 | 200 | 200 | 200 | 200 |
| 5 | 22 | 21 | 17 | 16 | 16 |
| 10 | 331 | 340 | 281 | 270 | 212 |
| 14 | 49 | 46 | 45 | 40 | 33 |
| 17 | 3824 | 3585 | 3303 | 3285 | 2085 |
| 26 | 2306 | 2195 | 1948 | 1904 | 937 |
| 29 | 5040 | 4977 | 4647 | 4354 | 2020 |
| 42 | 4159 | 4159 | 3869 | 3719 | 1273 |
| 48 | 135279 | 127276 | 121034 | 116964 | 10628 |
| 76 | 171510 | 179728 | 173609 | 179123 | 108159 |

Tabela 1 długości wyznaczonych tras

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

Tabela 2 czas działania algorytmu [ms]

Poniższe wykresy obrazują zależności pomiędzy liczbą miast, współczynnikiem chłodzenia i czasem wykonywania się algorytmu.

Na podstawie pierwszego wykresu można przyjąć, że czas rośnie liniowo wraz ze wzrostem liczby miast w grafie. Im większy współczynnik chłodzenia, tym więcej iteracji się wykona i czas się odpowiednio wydłuży.

Rysunek 1 wykres czasu od liczby miast dla różnych współczynników chłodzenia

Rysunek 2 wykres długości drogi od współczynnika chłodzenia dla różnych wielkości grafów

Drugi wykres przedstawia długość wyliczonej drogi w zależności od współczynnika chłodzenia dla danego grafu. Wraz ze wzrostem współczynnika, rośnie liczba iteracji, dzięki czemu możemy uzyskać krótszą drogę. Sprawdza się to w większości przypadków, gdyż poprzez zależność od losowego rozwiązania początkowego i losowych rozwiązań w poszczególnych iteracjach, może się zdarzyć, że wylosowana droga będzie dużo dłuższa i nie zostaną wylosowane takie pary, których zamiana sprawi skrócenie drogi względem krótszej, znalezionej w poprzednim teście i tym samym minimum lokalne będzie większe, niż minimum lokalne z poprzedniego testu.

Rysunek 3 wykres czasu od współczynnika chłodzenia dla różnych wielkości grafów

Na powyższym wykresie widać rosnący czas wraz ze wzrostem współczynnika chłodzenia, co jest znów następstwem wzrostu liczby iteracji wykonywania się algorytmu.

# Podsumowanie i wnioski

Analizując otrzymane wyniki można zauważyć pewne prawidłowości:

* Czas wykonywania się rośnie wraz z liczbą instancji. Dla małych grafów algorytm wykonuje się szybko.
* Czas wykonywania się rośnie wraz ze współczynnikiem chłodzenia przez wzrost liczby iteracji.
* Większa liczba iteracji pozwala na znalezienie rozwiązania bliższego optymalnemu, jednak wydłuża ona czas działania.
* Wraz ze wzrostem współczynnika chłodzenia możemy osiągnąć lepsze rozwiązania.

Powyższe zależności można pokrótce opisać następująco: zwiększając czas zyskujemy na jakości rozwiązania, zmniejszając go, tracimy.

# Bibliografia

1. Symulowane wyżarzanie: https://tdb0.wordpress.com/2010/12/03/symulowane-wyzarzanie/