Sprawozdanie - projekt 2.

Implementacja i analiza efektywności algorytmu
symulowanego wyżarzania dla problemu komiwojażera.

Autor: Michał Bańka, 235051
Prowadzący: mgr inż. Antoni Sterna
Termin zajęć: Czwartek, 7:30

1. OPIS PROBLEMU

Problem komiwojażera (ang. TSP – Travelling Salesman Problem) jest problemem, który polega na znalezieniu najkrótszego cyklu Hamiltona w grafie. Oznacza to, że ścieżka powinna zawierać wszystkie wierzchołki dokładnie jeden raz i przez każdą krawędź przechodzić również najwyżej jeden raz. Dodatkowo ostatni wierzchołek powinien być połączony z pierwszym, tak aby ścieżka stała się cyklem.

Dla 4 wierzchołków mamy 3*2*1 możliwości wyboru ścieżek, dla 5 wierzchołków 4*3*2*1 ścieżek itd. Tak więc dla *n* wierzchołków ilość ścieżek do sprawdzenia czy są najkrótszymi cyklami Hamiltona wyniesie (n-1)*(n-2)*…*2*1 = (n-1)! . Najprostsza metoda algorytmu rozwiązującego problem będzie miała złożoność (n-1)!.

Sprowadza to problem komiwojażera do zbioru problemów NP-trudnych, a właściwie do jego podzbioru problemów NP.-zupełnych. Oznacza to, że problem jest możliwy do sprawdzenia poprawności w czasie wielomianowym, ale nie jest możliwe znalezienie jego rozwiązania w czasie wielomianowym. Złożoność problemu komiwojażera jest wykładnicza, gdyż: $n!>2^n$ już od n=4.

2. OPIS ALGORYTMÓW

A. ALGORYTM SYMULOWANEGO WYŻARZANIA

To rodzaj algorytmu heurystycznego przeszukującego przestrzeń problemu w celu wyszukania możliwie najlepszych rozwiązań. Algorytm ten posiada swoją nazwę od wyżarzania metali. Wyżarzanie polega na rozgrzewaniu metalu tak, aby cząsteczki nabrały energii i zaczęły się szybciej poruszać. W przeciwieństwie do hartowania spadek w tym procesie spadek temperatury powinien być możliwe powolny tak aby cząsteczki znalazły odpowiednie miejsce w strukturze.

Podobnie zachowuje się algorytm z tą różnicą, że zamiast wyboru pozycji wybieramy możliwie najlepsze rozwiązanie, czyli w przypadku problemu komiwojażera wybieramy najkrótszy cykl.

Przebieg algorytmu zmienia swój charakter z czasem trwania. Początkowo, kiedy temperatura jest wysoka jest stosunkowo duża szansa na przejęcie rozwiązania znajdującego się blisko w przestrzeni rozwiązań, pozwala to na znalezienie otoczenia minimum lokalnego. Z czasem kiedy temperatura spada, rozwiązanie coraz bardziej zbliża się do minimum lokalnego i z coraz mniejszym prawdopodobieństwem się od niego oddala.

Algorytm symulowanego wyrażania jest rozszerzoną wersją przeszukiwania lokalnego. Różnica polega na tym, że istnieje pewne prawdopodobieństwo na ruch przeciwny do oczekiwanego, tzn. może poruszać się w stronę gorszego rozwiązania.

Pseudokod dla tego algorytmu wygląda następująco:

Start

Inicjalizuj($Cykl_{Początkowy}$, $Temperatura_{Początkowa}$, $Temperatura_{Minimalna}$, Długość_Schodka) ;

$Cykl_{aktualny} = Cykl_{Początkowy}$;

Wykonuj

Wykonuj

Generuj: $Cykl_{Sąsiad}$ na podstawie $Cykl_{aktualny}$

Jeżeli: Długość($Cykl_{Sąsiad} < Cykl_{aktualny}$)

$Cykl_{aktualny} = Cykl_{Sąsiad}$

W przeciwnym wypadku, jeżeli: $p[0,1) < e^{\frac{Dlugosc(Cykl_{sasiada}) - Dlugosc(Cykl_{aktualny})}{Temperatura_{aktualna}}}$

$Cykl_{aktualny} = Cykl_{Sąsiad}$

Dopóki: i < Długość_schodka

Wykonaj funkcję zmiany temperatury

Dopóki: $Temperatura_{Aktualna} > Temperatura_{Minimalna}$

Koniec

3. Opis implementacji algorytmów i struktur

a. Przechowywanie grafu

Graf przechowywany jest w stworzonej specjalnie dla niego klasie Matrix. Klasa ta została stworzona do przechowywania macierzy kwadratowych i wykonywania prostych operacji na niej. Dodatkowo na macierzy można zastosować opisywane w punkcie 2. algorytmy.

Struktura grafu zapisana jest w postaci macierzy sąsiedztwa, gdzie jako pierwsza współrzędna macierzy podaje się wierzchołek, z którego krawędź wychodzi, a jako druga wierzchołek końcowy krawędzi.

Wszystkie struktury alokują pamięć dynamicznie, co zostało uwzględnione podczas mierzenia czasu.

b. Implementacja algorytmu symulowanego wyżarzania

Algorytm symulowanego wyżarzania jest algorytmem, w którym na czas wykonywania oraz jakość znalezionego rozwiązania bardzo duży wpływ mają wartości inicjujące algorytm. Najważniejsze z nich to początkowy cykl, początkowa temperatura, długość „schodka" temperatury, funkcja obniżająca temperaturę, sposób generowania sąsiedztwa i temperatura minimalna.

W napisanym programie zainicjalizowane dane pozwalają na sprawne wykonywanie algorytmu nawet dla grafów pełnych o rozmiarze 171 wierzchołków. Jakość rozwiązania dla tak dużych grafów jest jednak stosunkowo słaba. Najlepiej sprawdza się on dla grafów o rozmiarze *n < 100.*

Początkowy cykl zdaje się mieć duże znaczenie dla wykonywania całego algorytmu, jednak znalezienie rozwiązania dającego, które zwiastuje znalezienie się w minimum globalnym zajmuje sporo czasu i dla tak dużych grafów może być skomplikowane.

Ważna jest wspomniana również temperatura. Początkowa temperatura w tym programie może być wyznaczana dwojako. Obie funkcje opierają się o wyszukiwanie n losowych cykli oraz sąsiadów dla każdego z nich. Pierwsza z nich wyciąga średnią ze wszystkich różnic odległości cyklu i sąsiada. Druga zwraca maksymalną różnicę tych długości. Jeśli wyznaczyć odpowiednio dużą ilość cykli, możemy zakładać maksymalną bądź średnią różnicę dla całego grafu. Pozwoli nam to określić prawdopodobieństwo z jakim pierwsze schodki będą przyjmować nowe rozwiązania.

Funkcja z maksymalną różnicą sprawi, że rozwiązania będą zmieniać się prawie ze stu procentową pewnością co pozwoli na znalezienie bardziej obiecującego rozwiązania przed następną zmianą temperatury. W programie nie zostało to zaimplementowane jednak takie rozwiązanie miało by sens gdyby przy każdej zmianie temperatury aktualnie sprawdzany cykl przyjmował dane z minimalnego znalezionego dotąd cyklu. Jako, że używana jest metoda ze średnią różnicą, algorytm jest szybszy, ponieważ szybciej osiąga temperaturę minimalną, jest też jednak minimalnie mniej dokładny.

Ustawienie minimalnej temperatury i funkcji jej zmian ma wpływ na jakość i czas wykonania. W projekcie funkcja zmiany jest funkcją wykładniczą: $\mathrm{temp.} = 0{,}95 * \mathrm{temp.}$ Minimalna temperatura została ustawiona na 0,001. Im wyższa temperatura początkowa tym dłużej będą trwały obliczenia.

Aby zapobiec długiej pracy programu, który utknie w minimum lokalnym i prawdopodobieństwo na wyjście z niego będzie bardzo mało, zaimplementowana została dodatkowa zmienna, która kończy działanie algorytmu jeśli przez ostatnie 750 operacji porównań sąsiadów żaden nie zostanie zamieniony z aktualnym rozwiązaniem.

Sposób generowania sąsiedztwa może odbywać się na 3 sposoby, losowane są 2 wierzchołki cyklu (nie jednakowe):

- *swap* – w tym sposobie te dwa wierzchołki w cyklu są zamieniane miejscami.
- *insert* – tutaj większy wylosowany wierzchołek zajmuje miejsce mniejszego, a wszytkie pozostałe są przesuwane o jeden dalej.
- *invert* – kolejność ciągu wszystkich wierzchołków pomiędzy pierwszym i drugim wylosowanym (włącznie z nimi) jest odwracana.

4. Plan eksperymentu

Plan eksperymentu zakłada wykonanie algorytmu po 50 razy dla gotowych macierzy wybranych ze strony internetowej podanej przez prowadzącego. Są to odpowiednio pliki z macierzami o rozmiarach: 17 (br17), 34 (ftv33), 54 (ftv55), 70 (ftv70) i 171 (ftv170). Zapisywany jest czas wykonywania algorytmu i długość znalezionego cyklu.

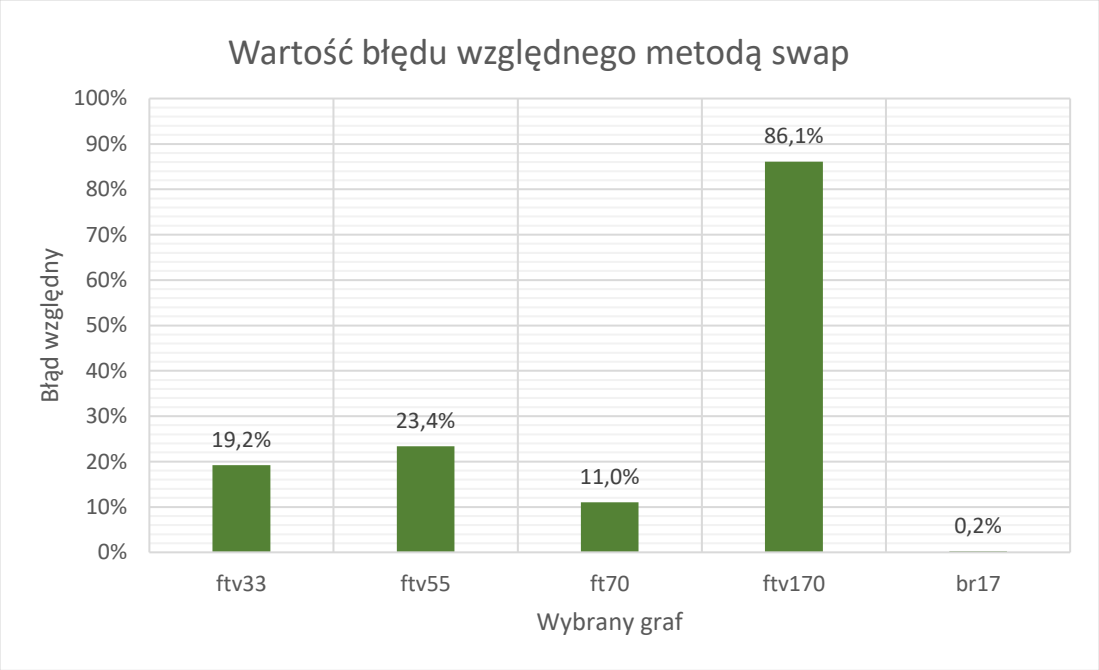
Dodatkowo dla grafów o rozmiarze 17 i 34 zostało wykonane porównanie metod generowania sąsiadów. Te rozmiary uruchamiane było po 50 razy dla każdego z trzech sposobów generowania.

Pozostałe rozmiary wykonywane były metodą *swap,* ze względu na ich dłuższy czas wykonywania i brak wyraźnych różnic w czasach działania algorytmu między różnymi metodami.

Pomiary czasu mierzą jedynie czas wykonania algorytmów, bez zmian elementów, wypisywania danych do konsoli i innych czynności wymaganych do obsługi programu.

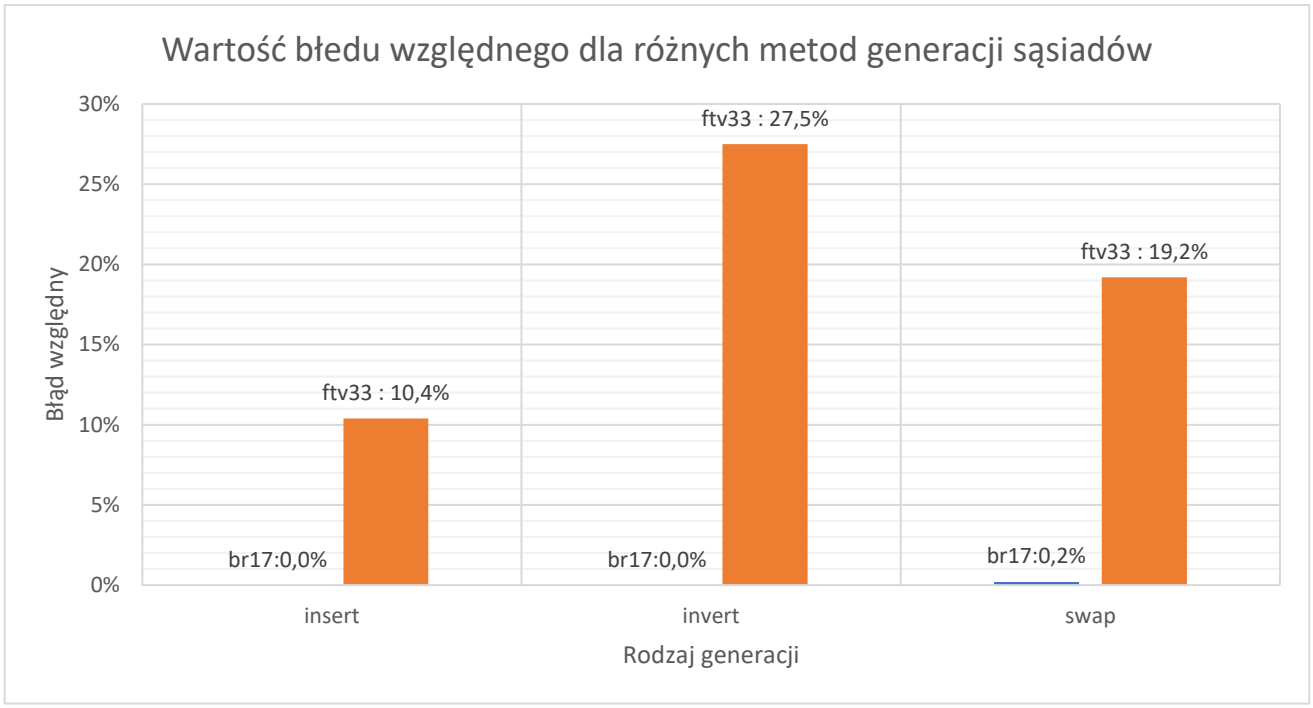
Pomiar czasu powinien odbywać się przez funkcję *QueryPerformanceCounter,* zapewniający stosunkowo wysoką jakość pomiarów rzędu mikrosekund.

5. Wyniki eksperymentów



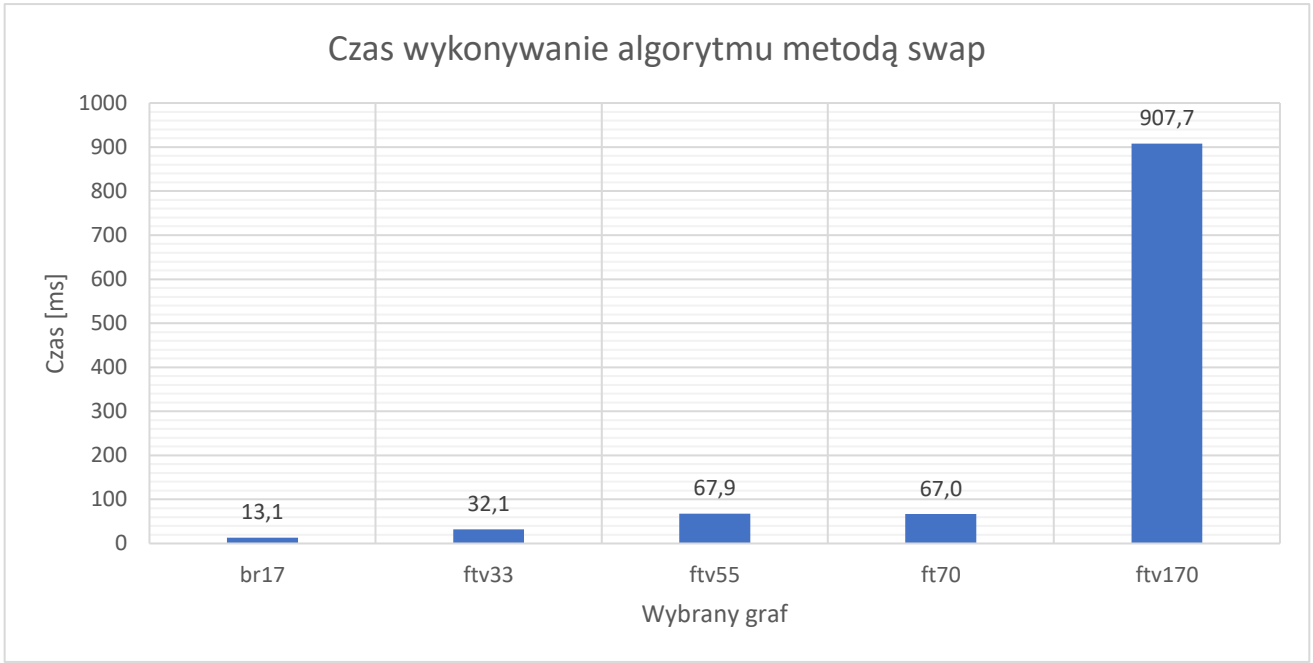
Wykres 1

	ftv33	ftv55	ft70	ftv170	br17
Błąd względny	19,2%	23,4%	11,0%	86,1%	0,2%



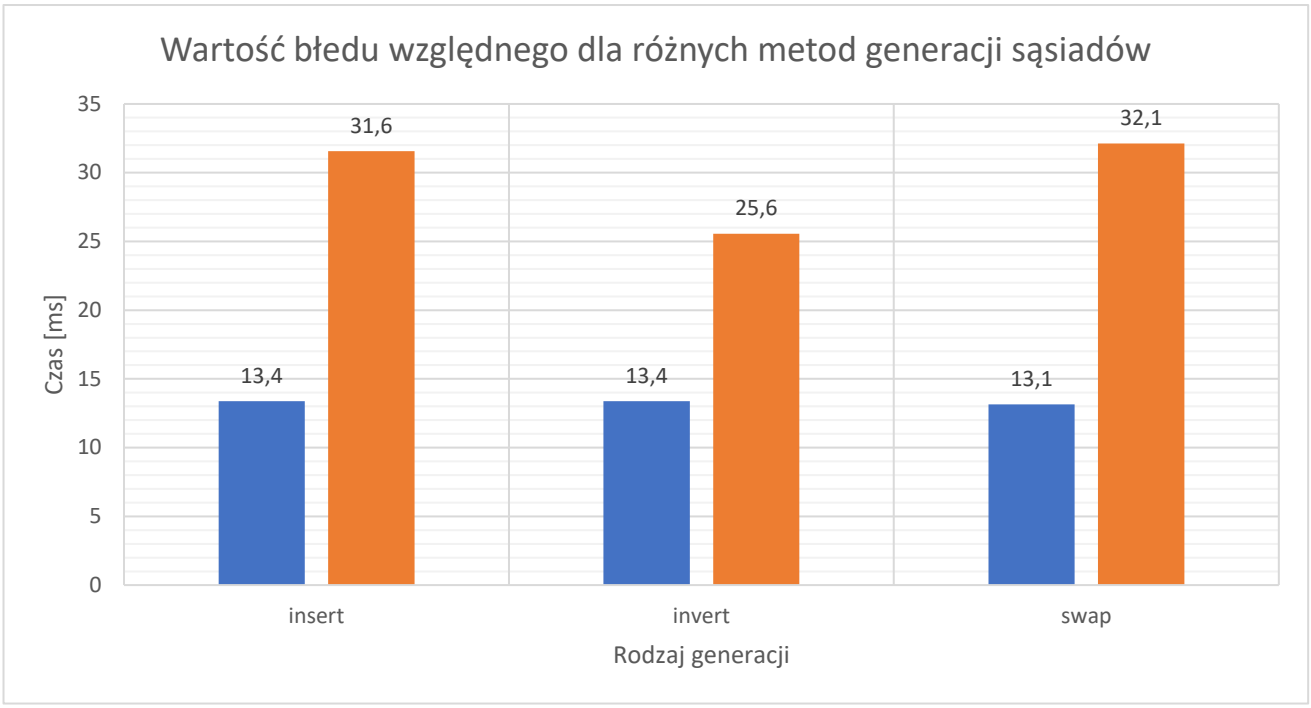
Wykres 2

		Rodzaj generowania		
		insert	invert	swap
Błąd względny	br17	0	0	0,002
	ftv33	0,104	0,275	0,192



Wykres 3

	br17	**ftv33**	**ftv55**	**ft70**	**ftv170**
Czas [ms]	13,1	32,1	67,9	67,0	907,7



		Rodzaj generowania		
		insert	invert	swap
Błąd względny	br17	13,37	13,37	13,15
	ftv33	31,56	25,57	32,12

6. Wnioski

Pierwszym najbardziej widocznym na wykresach wnioskiem jest brak błędu względnego i prawie zerowa różnica wykonywania algorytmu między różnymi sposobami generacji dla grafu o 17 wierzchołkach. Spowodowane jest to przede wszystkim specyficznymi wartościami przechowywanymi w grafie. Wartości krawędzi często się powtarzają co tworzy bardzo szerokie minimum globalne. Algorytm z małym prawdopodobieństwem może nie wykryć minimum globalnego w tym grafie, co potwierdza metoda swap, gdzie występuje mała różnica.

Wielkość struktury ma wpływ na czas wykonywania algorytmu chociaż, nie jest to jedyny czynnik. Ważne są również same dane przechowywane w macierzy reprezentującej graf, tzn. im ścieżki między wierzchołkami są dłuższe tym algorytm jest dłużej wykonywany, ale za to dokładniej. Zgadza się to z przewidywaniami z punktu 3.

Jeśli chodzi o sposób generacji danych możemy zauważyć, że czas wykonywania algorytmu przez sąsiedztwo swap i insert jest podobne, ale to insert zwraca niemal dwukrotnie mniejszy błąd względny niż swap i trzykrotnie mniejszy niż invert. Invert to metoda, która trwa najkrócej, ale jest najmniej dokładna.

Dosyć dużym błędem i czasem wykonywania odznacza się macierz o rozmiarze 171.