Projektowanie Efektywnych Algorytmów

Projekt

17/10/2023

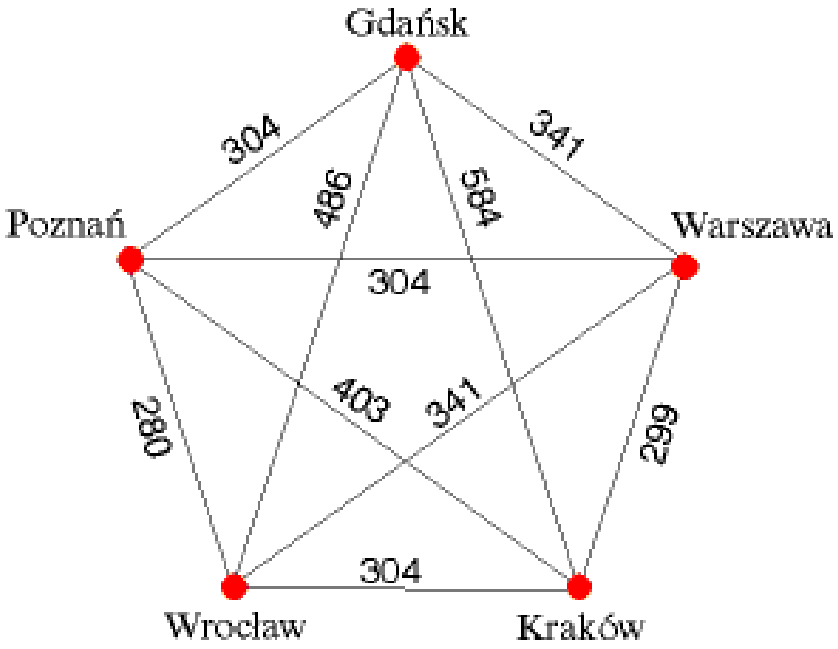
263896 Patryk Jurkiewicz

(1) Brute Force

Treść zadania	Strona
Sformułowanie zadania	1
Metoda	2
Algorytm	3
Dane testowe	4
Procedura badawcza	5
Wyniki	6
Analiza wyników i wnioski	7

1. Sformułowanie zadania

Zadanie polega na opracowaniu i implementacji algorytmu, który wykorzystuje metodę przeglądu zupełnego (brute force) do rozwiązania problemu komiwojażera. Problem komiwojażera polega na znalezieniu najkrótszej trasy, która odwiedza wszystkie miasta (wierzchołki) dokładnie raz, a następnie wraca do miasta początkowego. To zadanie jest istotne w wielu dziedzinach, takich jak logistyka, planowanie tras, a także w rozwoju algorytmów optymalizacyjnych. Naszym celem jest więc opracowanie aplikacji, która samodzielnie rozwiąże ten problem za pomocą algorytmu opartego na przeglądzie zupełnym. Program ten ma służyć wyłącznie do analizy i rozwiązania problemu komiwojażera z wykorzystaniem naszego własnego algorytmu, opartego na metodzie przeglądu zupełnego, i dostarczyć pełny opis tego algorytmu w raporcie końcowym.



Rys 1. Przykład grafu przedstawiającego problem Komiwojażera.

2. Metoda

Metoda przeglądu zupełnego, nazywana także przeszukiwaniem wyczerpującym lub metodą siłową (brute force), jest podejściem do rozwiązywania problemów optymalizacyjnych, które polega na sprawdzaniu wszystkich możliwych rozwiązań dopuszczalnych w danym problemie. W ramach tej metody, analizowane są wszystkie potencjalne kombinacje i permutacje rozwiązań, a dla każdej z nich obliczane są wartości funkcji celu. Następnie, wybierane jest rozwiązanie, które osiąga ekstremalną wartość funkcji celu, czy to minimalną w przypadku problemów minimalizacyjnych, czy to maksymalną w przypadku problemów maksymalizacyjnych.

Metoda przeglądu zupełnego jest niezwykle dokładna, ponieważ bada wszystkie możliwe opcje, ale jest też bardzo kosztowna obliczeniowo. Przy dużych instancjach problemów może być praktycznie niewykonalna ze względu na eksplozywny wzrost liczby możliwych kombinacji. Pomimo swojej wydajności, metoda ta stanowi pewnego rodzaju "ostatnią deskę ratunku" w przypadku, gdy inne, bardziej efektywne algorytmy nie są dostępne lub nie dają satysfakcjonujących wyników.

W praktyce, przy rozwiązywaniu problemu komiwojażera za pomocą metody przeglądu zupełnego, wszyscy możliwi klienci i trasy są analizowane, a najlepsza trasa jest wybierana na podstawie minimalizacji całkowitej długości trasy.

3. Algorytm

Część 1: Inicjalizacja i Wczytywanie Danych

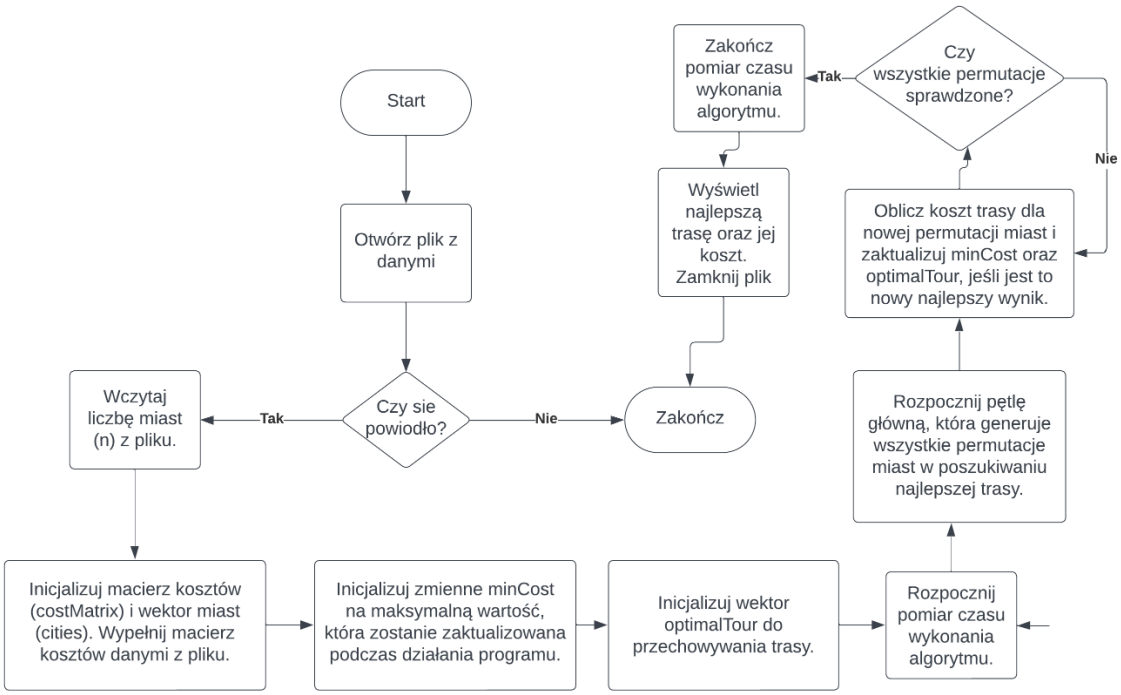
Program rozpoczyna się od otwarcia pliku "dane.txt" z danymi. Po wczytaniu liczby miast, inicjalizowane są macierz kosztów (costMatrix) i wektor miast (cities), a także zmienne takie jak minCost. Algorytm rozpoczyna pomiar czasu.

Część 2: Generowanie i Ocena Permutacji

W tej części algorytmu uruchamiana jest główna pętla. Jej zadaniem jest generowanie wszystkich możliwych permutacji miast w poszukiwaniu najlepszej trasy. Dla każdej permutacji, algorytm oblicza koszt trasy, porównuje go z dotychczasowym najlepszym wynikiem (minCost), i jeśli jest to nowy najlepszy wynik, aktualizuje minCost oraz wektor optimalTour. Po każdej iteracji generowana jest kolejna permutacja miast, a pętla kontynuuje swoje działanie aż do wyczerpania wszystkich możliwych permutacji.

Część 3: Wyświetlanie Wyników i Zakończenie Programu

Po zakończeniu generowania i oceny permutacji, algorytm zatrzymuje pomiar czasu, wyświetla najlepszą trasę i jej koszt. Następnie zamyka plik z danymi i kończy działanie algorytmu komiwojażera.



4. Dane testowe

W celu przetestowania i dostrojenia naszego algorytmu wybraliśmy zestaw konkretnych instancji problemu komiwojażera. Te instancje posłużą nam do oceny skuteczności i poprawności działania algorytmu oraz do określenia odpowiednich parametrów.

Do wykonania badań wybrano następujący zestaw instancji ze strony: http://jaroslaw.mierzwa.staff.iiar.pwr.wroc.pl/pea-stud/tsp/

1. tsp_6_1.txt
2. tsp_6_2.txt
3. tsp_10.txt
4. tsp_12.txt
5. tsp_13.txt
6. tsp_14.txt

5. Dane testowe

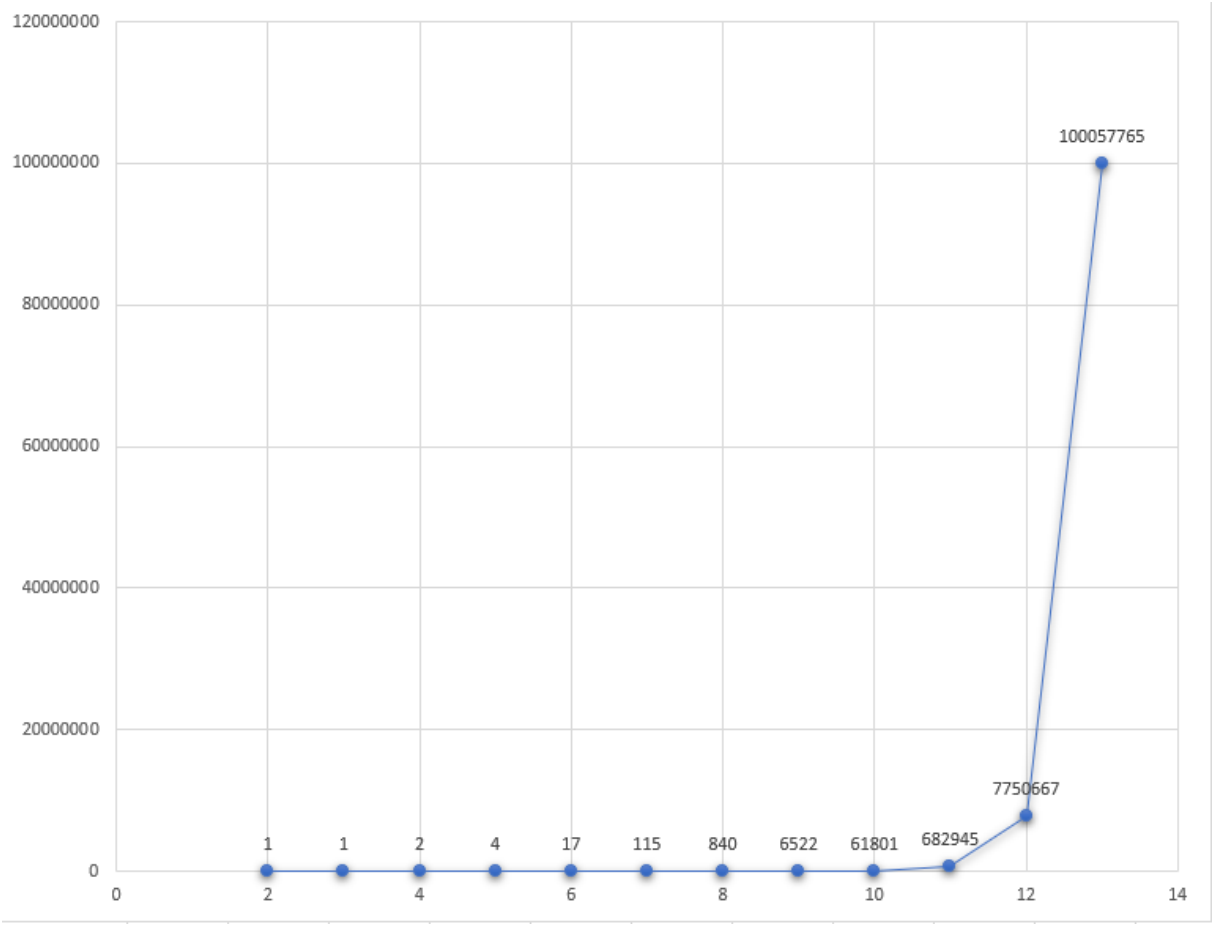
Aby wykonać ten proces, uruchom plik wykonywalny (exe) i wprowadź liczbę iteracji na początku działania programu. Wyniki zostaną zapisane w pliku "wyjscie.txt" w formacie, który przypomina strukturę przedstawioną w poniżej:

Najlepsza trasa (numer iteracji): trasa
Koszt najkrótszej trasy (numer iteracji): czas

Dane wczytywane są z podanego na początku pliku tekstowego.

6. Wyniki

Wyniki zostały zgromadzone w plikach o formacie „wyjście(ilość miast).txt” Wyniki przedstawione zostały w postaci wykresu zależności czasu uzyskania rozwiązania problemu od wielkości instancji (rysunek 2).

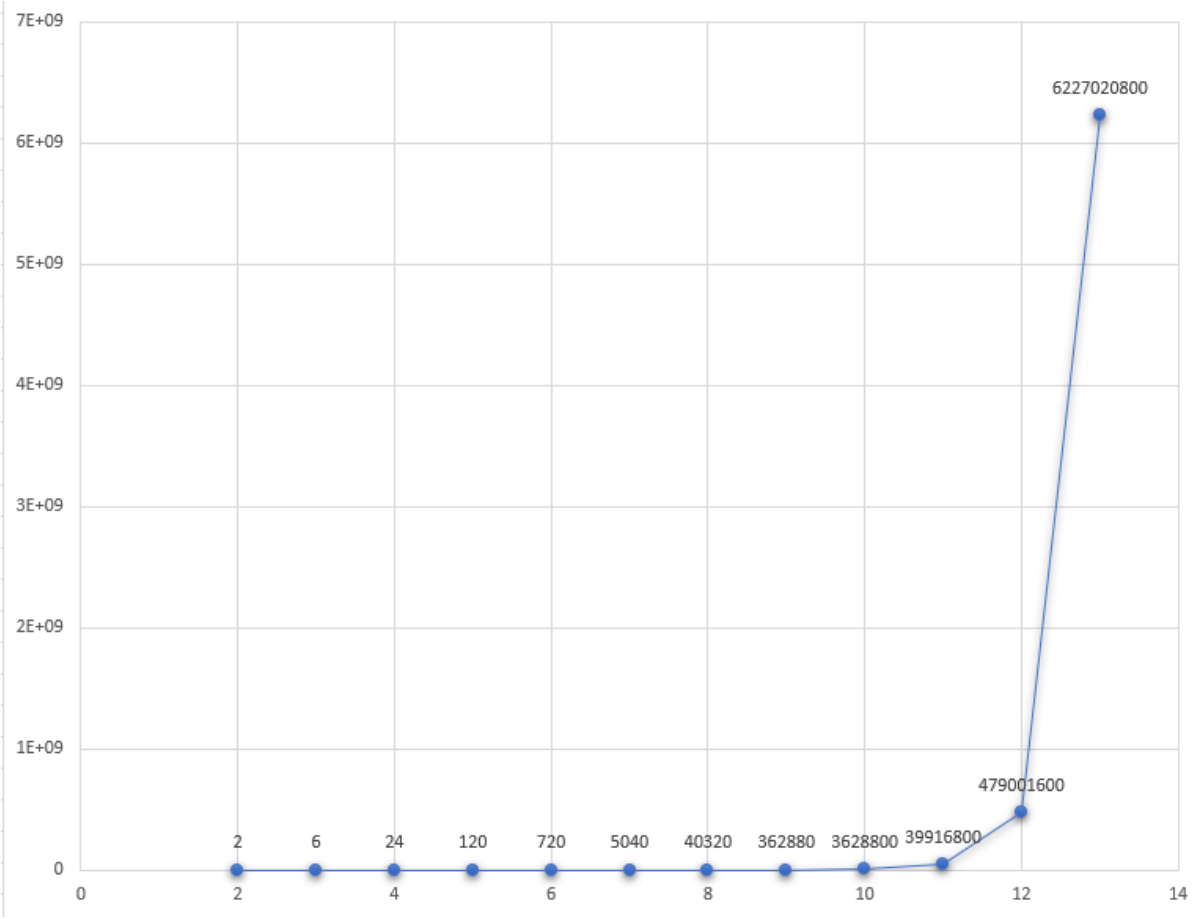


Rys 2. Wykres przedstawiający ilość mikrosekund potrzebnych na poradzenie sobie z problemem komiwojażera w zależności od liczby miast.

7. Analiza wyników

Analizując prezentowane dane (rys 2.), można dostrzec, że średni czas rozwiązania problemu komiwojażera znacząco wzrasta wraz z liczbą miast. To sugeruje, że trudność tego zadania rośnie wykładniczo w zależności od liczby miast, a krzywa wzrostu czasu ma wyraźny charakter wykładniczy.

Co istotne, nałożenie krzywej czasu na krzywą reprezentującą funkcję n! (silnia) wskazuje, że algorytm użyty do rozwiązywania problemu komiwojażera działa w czasie, który jest ściśle związany z funkcją silni. W praktyce oznacza to, że dla każdej nowej instancji problemu, czas potrzebny na jego rozwiązanie rośnie nieliniowo w stosunku do liczby miast, co sprawia, że rozwiązywanie problemu komiwojażera dla dużych ilości miast staje się wyjątkowo trudne i czasochłonne. Dla porównania załączam wykres silni(rys 3.), który mocno przypomina poprzedni wykres.



Rys 3. Wykres przedstawiający wynik silni kolejnych liczb naturalnych.