**PROJEKTOWANIE EFEKTYWNYCH ALGORYTMÓW**

PROJEKT

07/10/2021

252736 Hutnik Szymon

Branch & Bound (3)

|  |  |
| --- | --- |
| Strona | Spis treści |
| 2 | Treść zadania |
| 3 | Opis metody |
| 4 | Opis algorytmu |
| 6 | Dane testowe |
| 7 | Procedura badawcza |
| 9 | Wyniki |
| 10 | Analiza |

1. Treść zadania

Opracować, napisać, zbadać rozwiązanie problemu komiwojażera w wersji optymalizacyjnej algorytmem *Branch & Bound*.

Problem komiwojażera (eng. *Travelling Salesman Problem*) polega na znalezieniu minimalnego cyklu Hamiltona (przejście przez wszystkie wierzchołki tylko raz, startując i kończąc w tym samym punkcie) w pełnym grafie ważonym.

1. Opis metody

Metoda podziału i ograniczeń (eng. *Branch & Bound*) polega analizie drzewa rozwiązań, które obrazuje kazdy możliwy sposób otrzymania rozwiązania końcowego. Przejście i wygenerowanie całego drzewa jest bardzo kosztowne, więc ogranicza się je w wybrany sposób. Wyróżniamy 3 strategie przeszukujące drzewo:

* Breadth search – przeszukiwanie drzewa w szerz
* Depth search – przeszukiwanie drzewa w głąb
* Min/Max Cost – Best Search

W projekcie zbadam wariant Best Search, ponieważ pozostałe 2 są skrajnie nieoptymalne nawet w porównaniu do algorytmu Brute Force ze względu na ogromną złożoność pamięciową około O(). W przypadku wykorzystania metody Best Search nie przechodzimy po wszystkich wierzchołkach drzewa, ale korzystając z kolejki priorytetowej wybieramy najbardziej obiecującą ścieżkę. Następnie po sprawdzeniu i nowych ścieżek do kolejki ponownie wybieramy najlepszy element i powtarzamy, aż do skutku.

1. Opis algorytmu

Rozwiązanie zaimplementowano w postaci programu opisanego przez poniższe diagramy:

Diagram

Description automatically generated

Rysunek 1: Ogólny diagram czynności programu

Najpierw inicjalizowane są zmienne, najlepsza ścieżka jest ustawiona na maksymalną wartość. Po wczytaniu danych z konsoli następuje uruchomienie właściwej części algorytmu, następnie wypisywany jest wynik oraz czas wykonania właściwego algorytmu.

Diagram

Description automatically generated

Rysunek 2: Szczegółowy diagram czynności algorytmu

1. Dane testowe

Dane, na których była badana efektywność algorytmu pochodzą ze zbioru udostępnionego przez dr Rudego. Do badania użyto wartości z następujących plików:

* m14.astp
* m15.astp
* ulysses16.tsp
* gr17.tsp
* gr21.tsp
* ulysses22.tsp
* gr24.tsp

http://jaroslaw.rudy.staff.iiar.pwr.wroc.pl/files/pea/instances.zip

1. Procedura badawcza

Należało zbadać zależność czasu rozwiązania problemu od wielkości instancji. W przypadku algorytmu realizującego przegląd zupełny przestrzeni rozwiązań dopuszczalnych nie występowały parametry programu, które mogły mieć wpływ na czas i jakość uzyskanego wyniku. W związku z tym procedura badawcza polegała na uruchomieniu programu i wklejeniu do niego danych z plików wybranych do badania.

Każda z instancji została wykonana do 10 razy, aby uśrednić czasy, limitem okazało się n=22, mimo usuwania poprzednich wyników zajęta pamięć (według Visual Studio 2019) przekroczyła 19GB, test został przerwany. Wyniki były zapisywane w Excelu, następnie na ich podstawie została przeprowadzona analiza.

Pomiar czasu został wykonany przy użyciu bilbioteki chrono. Po otrzymaniu wyniku należy go podzielić przez liczbę powtórzeń wywołań algorytmu.

Text

Description automatically generated



1. Wyniki

Wyniki opracowano w programie Excel:

Rysunek 3: Czasy działania algorytmu dla *n* wierzchołków - graf



Rysunek 4: Czasy działania algorytmu dla *n* wierzchołków - tabela

Rysunek 5: Zużycie pamięci algorytmu dla *n* wierzchołków - graf

Rysunek 6: Porównanie czasu działania dotychczas opracowanych algorytmów dla *n* wierzchołków - graf

\*dane testowe nie są kolejnymi wartościami n stąd połączenie znanych wyników linią

1. Analiza

Krzywa wzrostu czasu względem wielkości instancji ma charakter losowy (Rysunek 3). Wynika to z faktu, że algorytm B&B ma złożoność czasową od O(*n*) do O(*n!*), tak niska złożoność wynika z faktu, że jeśli jedna ścieżka będzie tańsza od kosztu wejścia do każdego innego pierwszego wierzchołka niż wybrany to zrealizujemy tylko ją i będziemy mieć pewność, że otrzymaliśmy najlepszy wynik. Widać to na wykresach, gdzie czas wykonania *n*=17 jest dużo wyższy niż dla *n*=21. Warto zwrócić uwagę, że nawet wykorzystując dynamiczne struktury danych użyteczność algorytmu jest ograniczona, ponieważ złożoność pamięciowa wynosi od O(*n\*n*2) do O(*n!\*n*2).

Porównanie dotychczas zrealizowanych algorytmów pokazuje jak dużą przewagę ma podejście dynamiczne nad siłowym. Czas wykonania Helda-Karpa zauważalnie niższy jednak jest to okupione wysokim zużyciem pamięci. W przypadku B&B mamy szansę na ogromną poprawę czasu, ale również większą szansę na przekroczenie pamięci, co w moim odczuciu stawia to podejście poniżej algorytmu Helda-Karpa ze względu na jego zawodność.