Projektowanie Efektywnych Algorytmów

Projekt

16/11/2022

numer indeksu	imię i nazwisko
259190	Jędrzej Czykier

(2) Held-Karp

Spis treści strona

1. Sformułowanie zadania

Zadanie polega na opracowaniu, implementacji i zbadaniu efektywności algorytmów rozwiązujących problem komiwojażera, nazywanego również TSP *(eng. Traveling Salesman Problem)* w wersji optymalizującej. Problem ten, polega na znalezieniu minimalnego cyklu Hamiltona czyli znając zbiór wierzchołków grafu oraz wagi połączeń między nimi, należy znaleźć drogę w której każdy wierzchołek jest odwiedzany tylko raz, z wyjątkiem pierwszego do którego algorytm powinien wrócić na końcu działania.

Problem ten jest NP-trudny, co oznacza że jego rozwiązanie jest co najmniej tak trudne, jak rozwiązanie innych problemów należących do klasy NP. Przykładami innych problemów należących do tej klasy są między innymi: problem plecakowy i problem zbioru niezależnego.

Jednym ze sposobów rozwiązania TSP jest metoda Brute-force. Algorytm ten sprawdza po kolei wszystkie możliwości co czyni go relatywnie prostym do implementacji lecz powolnym w działaniu. Kolejnym jest algorytm Helda-Karpa, nazywany też algorytmem Bellmana-Helda-Karpa. Wykorzystuje on programowanie dynamiczne aby rozwiązać problem komiwojażera.

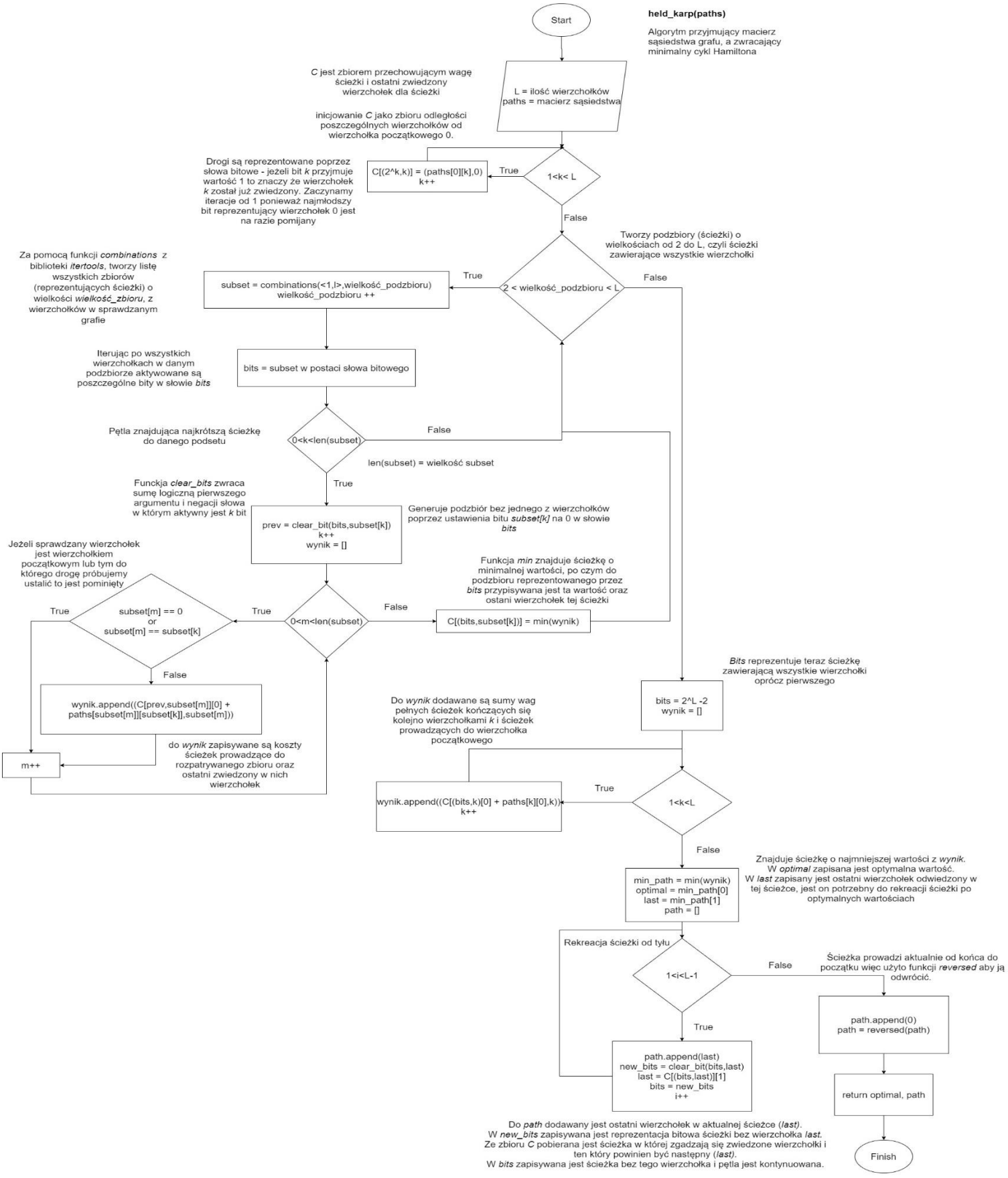
2. Metoda

Metoda Helda-Karp korzysta z techniki programowania dynamicznego czyli dzielenia zadania na mniejsze podzadania, których wyniki są wykorzystywane do otrzymania rozwiązania oryginalnego zadania. W przypadku problemu komiwojażera, zadaniem jest znalezienie minimalnego cyklu Hamiltona czyli dla danego zbioru wierzchołków *S* i krawędzi *k* gdzie ($1 \notin S, k \in S$) – znaleźć ścieżkę o minimalnej wartości poczynając od wierzchołka 1, przechodzącą przez wszystkie wierzchołki w *S*. Zadanie to jest dzielone na podzadania polegające na znalezieniu ścieżki wychodzącej z wierzchołka 1 a kończącej się na wierzchołku *m* gdzie ($m \in S, m \neq k$), przechodzącej przez wszystkie wierzchołki należące do *S* oprócz tych należących do podzbioru *k.* Po znalezieniu wszystkich takich ścieżek, sprawdzamy które można połączyć z powrotem z wierzchołkiem 1. Ścieżka o najmniejszej wadze (w przypadku algorytmu minimalizującego) jest naszym wynikiem.

Problem tej metody polega na tym, że przez to że musimy zapamiętać wszystkie kombinacje przejrzanych wierzchołków, używa dużych ilości pamięci. Sposobem aby to trochę zanegować jest reprezentacja zbiorów za pomocą, na przykład 32 bitowych słów INT. Zakładamy wtedy, że każdy bit reprezentuje poszczególny wierzchołek, który jeżeli został zwiedzony, ma ustawioną wartość na 1.

Algorytm Helda-Karpa jest dokładny, co oznacza że zawsze zwróci rozwiązanie optymalne.

3. Algorytm



Rysunek 1 Przedstawienie algorytmu za pomocą schematu blokowego

4. Dane testowe

Dane wykorzystane do sprawdzenia poprawności algorytmu oraz poprawne wyniki:

Nazwa pliku, optymalna ścieżka, waga optymalnej ścieżki

1. tsp_6_1.txt, [0, 1, 2, 3, 4, 5], 132,

2. tsp_6_2.txt , [0, 5, 1, 2, 3, 4], 80,

3. tsp_13.txt, [0, 12, 1, 8, 4, 6, 2, 11, 9, 7, 5, 3, 10], 269

Powyższe pliki można znaleźć po adresem:

http://jaroslaw.mierzwa.staff.iiar.pwr.wroc.pl/pea-stud/tsp/

Dane wykorzystane do badań:

Nazwa pliku, optymalna ścieżka, waga optymalnej ścieżki

1. tsp_6_1.txt, [0, 1, 2, 3, 4, 5], 132

2. tsp_10.txt, [0, 3, 4, 2, 8, 7, 6, 9, 1, 5], 212

3. tsp_13.txt, [0, 10, 3, 5, 7, 9, 11, 2, 6, 4, 8, 1, 12], 269

4. tsp_15.txt, [0, 10, 3, 5, 7, 9, 13, 11, 2, 6, 4, 8, 14, 1, 12],291

5. tsp_17.txt, [0,11,13,2,9,10,1,12,15,14,5,6,3,4,7,8,16], 39

6. gr21.txt, [0, 11, 3, 10, 19, 18, 16, 9, 17, 12, 13, 14, 20, 1, 2, 8, 4, 15, 5, 7, 6], 2707

Powyższe pliki można znaleźć pod adresami:

http://jaroslaw.mierzwa.staff.iiar.pwr.wroc.pl/pea-stud/tsp/

http://elib.zib.de/pub/mp-testdata/tsp/tsplib/tsp/index.html

Na potrzebę testów, plik gr21.txt został edytowany aby opisane w nich wartości były w postaci macierzy sąsiedztwa.

5. Procedura badawcza

Należało zbadać zależność czasu rozwiązania problemu oraz ilości użytej na rozwiązanie pamięci od wielkości instancji. W przypadku algorytmu Helda-Karpa w 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 sterowanego plikiem inicjującym .INI

Treść pliku .ini:

[section_a]

file0 = tsp_6_1.txt 50

file1 = tsp_10.txt 30

file2 = tsp_13.txt 25

file3 = tsp_15.txt 15

file4 = tsp_17.txt 10

file5 = gr21.txt 5

outputFile = outputHeldHerp.csv

Każda z instancji rozwiązywana była zgodnie z liczbą jej wykonań, np. tsp_6_1.txt wykonana została 50 razy. Do pliku wyjściowego outputHeldKarp.csv zapisywany był czas wykonania się algorytmu, otrzymane rozwiązanie (koszt ścieżki), ścieżka (numery kolejnych węzłów) oraz ilość pamięci użytej przy wykonywaniu się algorytmu. Plik wyjściowy zapisywany był w formacie csv.

Czas wykonywania się algorytmu był mierzony za pomocą funkcji z biblioteki *time*, a zużyta pamięć za pomocą funkcji z biblioteki *tracemalloc.* Obie biblioteki wbudowane są w język Python.

Poniżej przedstawiono fragment zawartości pliku wyjściowego.

tsp6_1.txt,"(132, [0, 1, 2, 3, 4, 5])",0.0009989738464355469,0.00336456298828125

tsp6_1.txt,"(132, [0, 1, 2, 3, 4, 5])",0.0,0.00336456298828125

…

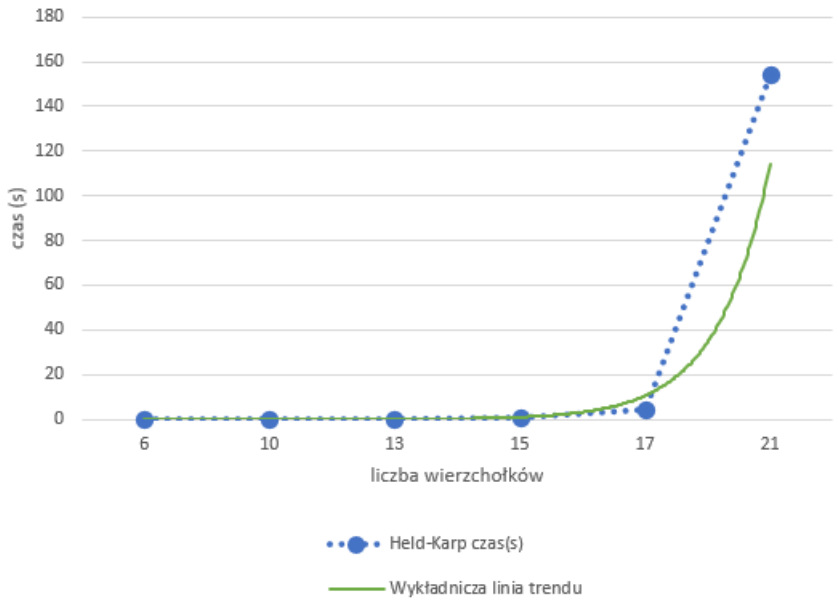
tsp_15.txt,"(291, [0, 12, 1, 14, 8, 4, 6, 2, 11, 13, 9, 7, 5,3,10])",0.88620519638,17.65126800537

6. Wyniki

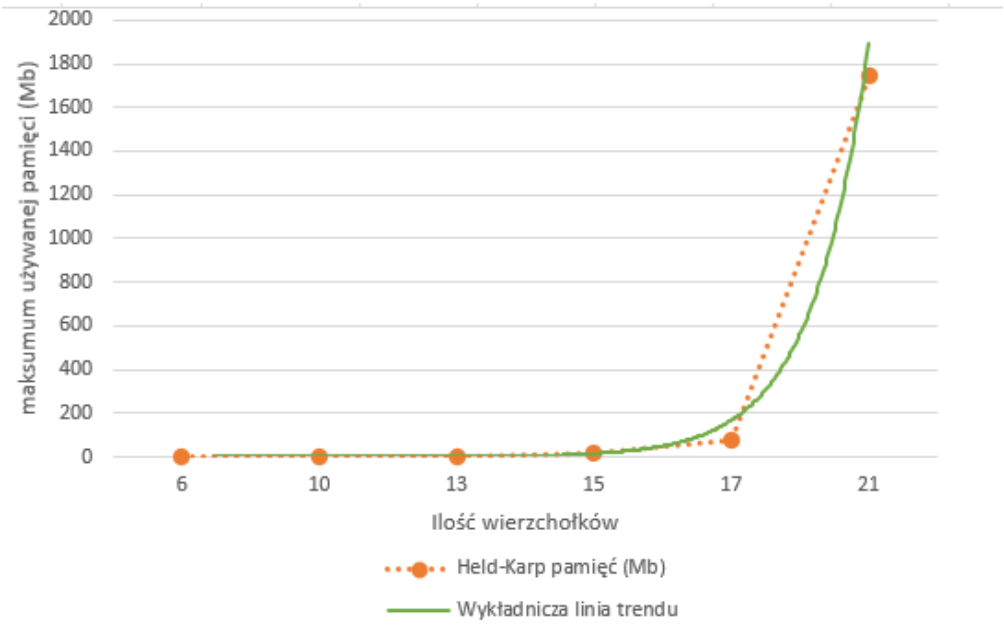
Wyniki zgromadzone w pliku outputHeldKarp.csv. Wszystkie pliki wynikowe zostały przesłane na Eportal.

Przy testowaniu algorytmu dla grafu o 24 wierzchołkach, program wykonywał się ponad 30 minut, więc przyjęto że granica możliwości sprzętu lub algorytmu znajduje się w grafach o 21 wierzchołkach.

Wyniki zostały przedstawione w postaci wykresów zależności czasu uzyskania rozwiązania problemu od wielkości instancji (Rysunek 2) oraz maksymalnej ilości użytej pamięci podasz rozwiązywania problemu zależnej również od wielkości instancji (Rysunek 3).



Rysunek 2 Zależność czasu rozwiązania problemu od wielkości instancji



Rysunek 3 Zależność maksymalnej ilości użytej pamięci podczas wykonywania się algorytmu

7. Analiza wyników i wnioski

Krzywe wzrostu czasu oraz używanej pamięci względem wielkości instancji mają charakter wykładniczy (rysunek 2) (rysunek 3). Złożoność czasowa opracowanego algorytmu wynosi $O(2^n n^2)$, a pamięciowa $O(n2^n)$. Jeżeli przyjmiemy że maksymalny czas jaki jesteśmy wstanie czekać na wyniki wynosi 30 minut to maksymalna ilość wierzchołków w rozpatrywanym grafie leży w przedziale {22,24} (dla sprzętu wykorzystanego do testów), ponieważ w grafie o 21 wierzchołkach, algorytm wykonał się w dopuszczalnym czasie lecz w grafie o 24 wierzchołkach, czas ten został przekroczony.