Politechnika Wrocławska
17.10.2019r

Projektowanie Efektywnych Algorytmów

Temat:
Rozwiązanie TSP za pomocą metod przeglądu zupełnego, programowania dynamicznego i metodą
podziału i ograniczeń

Prowadzący
dr. inż. Tomasz Kapłon

Autor
Paweł Parczyk

Ocena:

Wstęp

Początki problemu Komiwojażera (TSP - Travelling salesman problem) sięgają XIX wieku, kiedy to William Rowan Hamilton opisał problem istnienia cyklu n-elementowego w grafie o n wierzchołkach. Za pierwszego autora, który sformalizował matematycznie problem komiwojażera, uznaje się Karla Mengera, który w 1930 zwrócił szczególną uwagę na trudność w obliczeniu optymalnego rozwiązania. Z uwagi na prostotę opisu problemu i trudność optymalizacji algorytmów służących do jego rozwiązania, problem stał się bardzo popularny.

Celem projektu było stworzenie kilku algorytmów rozwiązujących TSP, a następnie określenie ich złożoności czasowych i pamięciowych na podstawie przeprowadzonych testów, a także porównanie różnych podejść pod względem szybkości, zapotrzebowania zasobów i skuteczności wyszukiwania optymalnego rozwiązania.

W ramach projektu zostały zaimplementowane cztery algorytmy: metoda przeszukiwania zupełnego (brute force), metoda programowania dynamicznego (algorytm Helda-Karpa), metoda zachłanna oraz metoda B&B.

Program został napisany w języku c++ w wersji 11 z wykorzystaniem technik programowania obiektowego. Kod został skompilowany przez kompilator g++, a następnie uruchomiony na komputerze zarządzanym przez system Linux Ubuntu 16.04.

Algorytmy zostały napisane w wersjach iteracyjnych w celu zminimalizowania czasu wykonania. Wersje rekurencyjne okazały się być bardzo powolne przez potrzebę wielokrotnego wywoływania funkcji (odkładanie ramek funkcji na stos programu).

Opis algorytmów

1. Metoda przeglądu zupełnego

Najprostsza i jednocześnie najbardziej intuicyjna metoda polegająca na sprawdzeniu wszystkich możliwych permutacji wierzchołków w grafie i wyboru takiego, w którym sumaryczny koszt odwiedzenia wszystkich wierzchołków jest najmniejszy. Algorytm ten ma bardzo wysoką złożoność czasową, ale jest bardzo prosty do implementacji. Jednocześnie gwarantuje znalezienie optymalnego rozwiązania.

Tak duża złożoność czasowa jest spowodowana bardzo szybkim przyrostem permutacji do rozpatrzenia. Ilość przypadków wyraża się wzorem $\Pi = n!$, co powoduje bardzo szybki ich przyrost. To sprawia że dla dużych grafów algorytm ten staje się bezużyteczny, a problemy nierozwiązywalne.

2. Metoda programowani dynamicznego

Metoda programowania dynamicznego cechuje się mniejszą złożonością czasową od metody przeglądu zupełnego, za to znacznie wyższą złożonością pamięciową. Polega ona na podzieleniu całego problemu na podproblemy i zapamiętywaniu wszystkich rozwiązanych podproblemów w celu późniejszego wykorzystania. Każdy podproblem może składać się z innych. Wiele podproblemów może mieć w sobie takie same podproblemy.
Algorytm Helda-Karpa opiera się o kolejne wywoływanie funkcji *H(V, d)*, gdzie *V* to zbiór wierzchołków $V=\{v_{1,}v_{2,}...v_n\}$, przez które trzeba dojść, a *d* $d\in s$ to wierzchołek docelowy, na którym należy zakończyć. Następnie po obliczeniu wartości funkcji jej wynik zostaje zapisany w ablicy i w przypadku, gdy ponownie zachodzi potrzeba wyliczenia wartości funkcji H, to korzysta się z tablicy kosztów.

Działanie algorytmu określa wzór rekurencyjny:
$H(\{v_1,v_2,...,v_n\},V_0)=min\{H(\{v_2,v_3,...v_n\},v_1)+D(v_{1,}V_0),H(\{v_1,v_3,...v_n\},v_2)$
$+D(v_{2,}v_0),...H(\{v_1,v_2,...v_{(n-1)}\},v_n)+D(v_n,v_0)\}$,

gdzie $H(\{v_i\},v_i)=D(v_{0,}v_i)$, a $D(v_i,v_j)$ to koszt dotarcia z wierzchołka v_i do wierzchołka v_j .

W programie wektor wierzchołków został zaprezentowany jako liczba Int, gdzie i-ty bit w tej liczbie oznacza czy i-ty+1 wierzchołek grafu zawiera się w V.

Przykład: *H(v, d)*, gdzie $s=(bin)0...000100101, d=(bin)0.0000000100$ oznacza, że należy policzyć koszt dotarcia z wierzchołka 0 do wierzchołka 3 (trzeci bit ustawiony na jeden) przechodząc przez wierzchołki 1, 3, 6. Zapis tak obliczonego kosztu odbywa się do tablicy kosztów *k[v][d]*.

Rozpatrując funkcje H od przypadków podstawowych tzn *rozmiar(V) = 1* można przerobić algorytm na wersję iteracyjną.

Zapis i odczyt z tablicy mają złożoność O(1) z uwagi na zastosowane indeksowanie.

3. Metoda zachłanna

Metoda zachłanna, wykorzystana z algorytmem najbliższego sąsiada, jest bardzo szybka i ma niewielka złożoność pamięciową, ale nie gwarantuje znalezienia optymalnego rozwiązania. Polega na przechodzeniu z wierzchołka zawsze do tego wierzchołka, do którego koszt przejścia jest najniższy.

4. Metoda przeglądu z ograniczeniami B&B

Metoda przeglądu z ograniczeniami jest modyfikacją metody przeglądu zupełnego. Z uwagi na pewne zależności pomiędzy kolejnymi rozwiązaniami można próbować wcześniej podzielić ścieżki dojścia do rozwiązania na takie, które dają nadzieję uzyskania lepszego rezultatu od znalezionego wcześniej i na takie, które nie dają takiej nadziej.

W algorytmie występują dwa ograniczenia górne i dolne. Górna granica jest obliczana jako koszt drogi przebytej do aktualnie przetwarzania wierzchołka. Dolna granica to z kolei najbardziej optymistyczna droga potrzebna do osiągnięcia wierzchołka przeznaczenia.

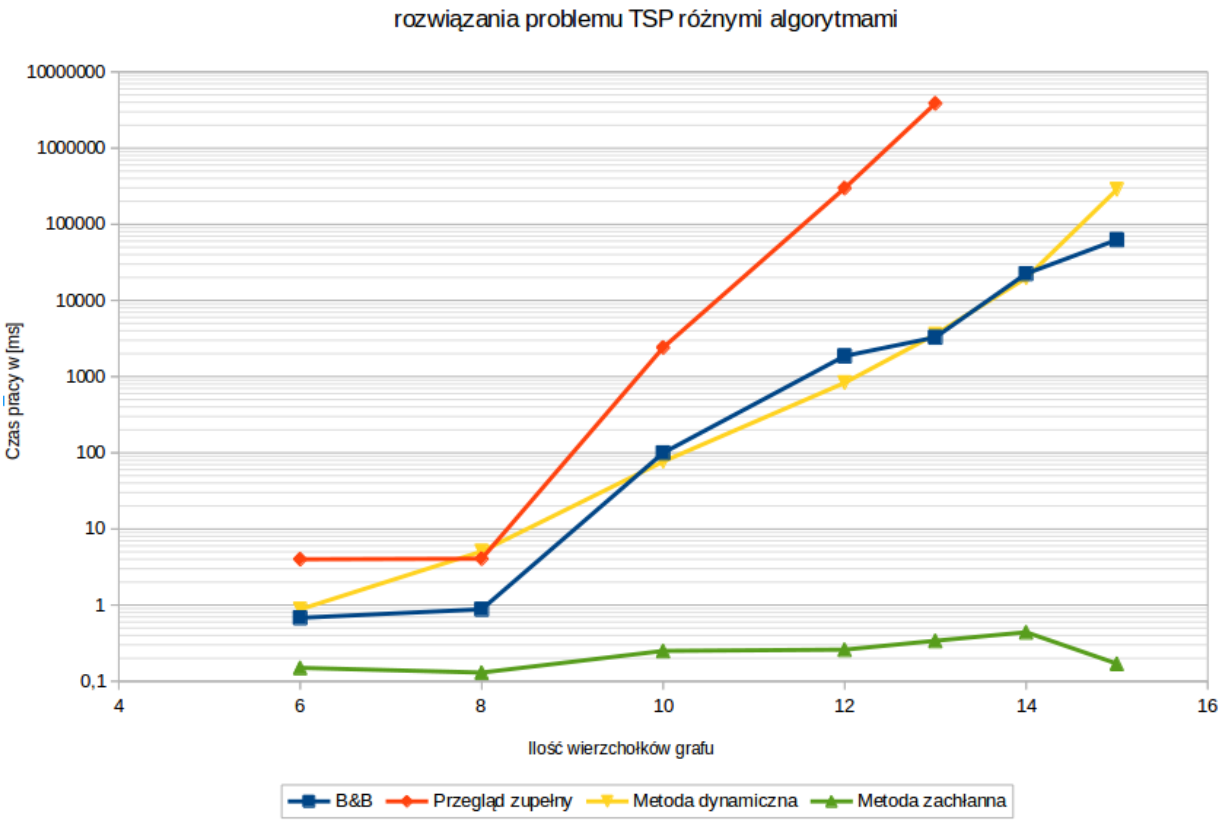
Podczas decyzji o rozwinięciu kolejnego wierzchołka sprawdzane są ograniczenia górne i dolne. Górne porównywane jest z kosztem dotarcia do kolejnego wierzchołka, i jeżeli ten koszt będzie większy od aktualnego minimalnego kosztu, to wierzchołek taki nie będzie rozwinięty. Podobnie badane jest ograniczenie dolne. Jeżeli koszt dotarcia do następnego wierzchołka, a następnie koszt dotarcia z niego do końca jest wyższy niż aktualnie najlepsze rozwiązanie, to wierzchołek taki nie zostanie rozwinięty.

Przebieg testów

Wszystkie algorytmy zostały przetestowane dla 7 grafów: dwóch 6-wierzchołkowych, 8-wierzchołkowego, 10-wierzchołkowego, 12-wierzchołkowego, 13-wierzchołkowego, 14-wierzchołkowego i 15-wierzchołkowego. Każdy pomiar został powtórzony 3 razy a wyniki zostały uśrednione. Poprawność algorytmów została określona na podstawie znanych odpowiedzi podanych w pliku *wyniki.as.*

Grafy i optymalne drogi do nich zostały pobrane ze strony dr inż. Jarosława Mierzwy.

Ilość wierzchołków	**Przegląd zupełny [ms]**	**Dynamiczne [ms]**	**Metoda zachłanna [ms]**	**B&B [ms]**
6(1)	0.86	0.961	0.25	0.85
6(2)	3.69	0.879	0.15	0.68
8	4.06	61.51	0.10	0.88
10	2406.84	76.65	0.25	99.60
12	298282.58	829.55	0.26	1864.80
13	38265241.74	3581.26	0.29	3310.59
14	>15h	19745.35	0.44	22450.10
15	>226h	286082.53	0.17	62526.20



Ilustracja 1: Porównanie czasów przetwarzania różnych algorytmów

2. Znalezione rozwiązania

Dla grafów dr Mierzwy zostały poniżej zapisane uzyskane rozwiązania:

- tsp_6: 0 – 1 - 2 – 3 - 4 – 5 – 0 dystans: 132,
- tsp_6_2: 0 – 5 – 1 – 2 – 3 – 4 – 0 dystans: 80,
- tsp_10: 0 – 3 – 1 – 9 – 6 – 7 – 8 – 2 – 4 – 5 – 0 dystans: 219 (graf ma więcej niż jedno rozwiązanie),
- tsp_12: 0 – 1 – 8 – 4 – 6 – 2 – 11 – 9 – 7 – 5 – 3 – 10 – 0 dystans: 264,
- tsp_14: 0 – 12 – 1 – 8 – 4 – 6 – 2 – 11 – 13 – 9 – 7 – 5 – 3 – 10 – 0 dystans: 282,
- tsp_15: 0 – 12 – 1 – 14 – 8 – 4 – 6 – 2 – 11 – 13 – 9 – 7 – 5 – 3 – 10 – 0 dystans: 291.

3. Przegląd zupełny

Zgodnie z oczekiwaniami metoda przeglądu zupełnego dobrze spisała się dla niewielkich grafów, gdzie znajdowała rozwiązania w czasie niewiele dłuższym od pozostałych metod. Jednak przy kolejnych grafach ilość permutacji wzrastała w bardzo szybkim tempie, co odbiło się na czasach. Przegląd zupełny ma złożoność O(n!), co powoduje, że duże instancje problemów stają się nierozwiązywalne przy jego pomocy. Graf o 12 wierzchołkach był analizowany przez 5 minut, a według oszacowania analiza grafu o zaledwie dwóch wierzchołkach więcej trwałaby ponad tydzień (około 226 godzin).

4. Metoda dynamiczna

Metoda programowania dynamicznego okazała się być bardzo szybką i skuteczną metodą. Czas analizy największego grafu przyjętego do testów zajął tylko niecałe 5 min, co jest bardzo

dobrym wynikiem porównując do metody przeglądu zupełnego. Złożoność obliczeniowa tej metody określa się na poziomie $O(n^2+2^n)$. Jednak z uwagi na złożoność pamięciową program bardzo szybko zapełnił całą dostępną pamięć. W przypadku pracy na komputerze przeznaczonym do testów 15 wierzchołków okazało się w sam raz. Program potrzebował około 8gb pamięci RAM. W przypadku rozwiązywania większych problemów system operacyjny wykorzystałby pamięć SWAP, co znacząco spowolniłoby działanie programu. Złożoność pamięciowa wynikająca z literatury wynosi $O(n\cdot 2^n)$.

5. Metoda zachłanna

Jest to jedyna metoda używa w testach, która nie daje pewności znalezienia optymalnego rozwiązania. Zgodnie z oczekiwaniami wraz ze wzrostem wielkości problemu czas realizacji nie zmienia się znacząco (w porównaniu do innych algorytmów). Złożoność obliczeniowa wynosi O(n).

6. Metoda B&B

Metoda B&B jako pewna modyfikacja przeglądu zupełnego ma taką samą ilość permutacji do przeanalizowania jak brute force, jednak dzięki określeniu pewnych ograniczeń znaczną cześć permutacji można odrzucić, przez co czas wykonania algorytmu znacznie się skraca. Jednak w najgorszym przypadku metoda B&B sprowadza się do przeglądu zupełnego (na przykład przy grafie w którym wszystkie wagi są jednakowe). Duży związek szybkości przetwarzania z wyglądem grafu można zaobserwować na podstawie wykresu.