Projektowanie Efektywnych Algorytmów

Projekt

18.11.2022

259062 Jacek Myjkowski

(2) BranchAndBound

Spis treści

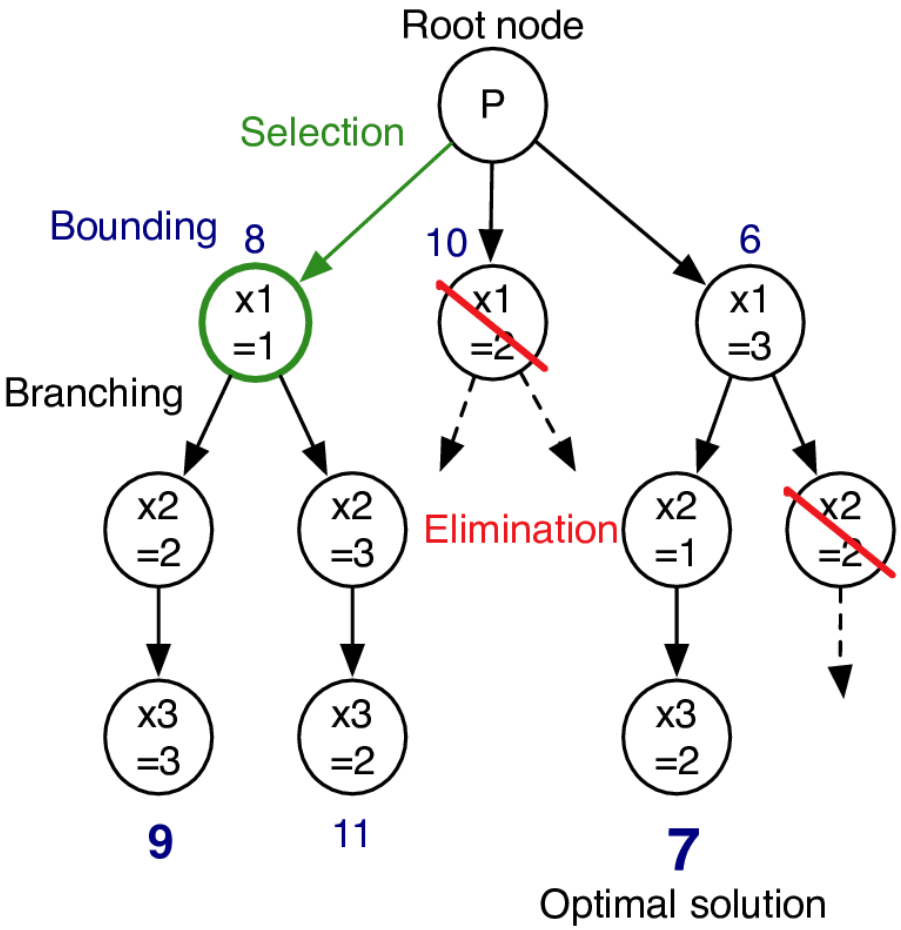
1. Sformułowanie zadania

Zadanie polega na opracowaniu, implementacji i zbadaniu efektywności algorytmu podziału i ograniczeń rozwiązującego problem komiwojażera w wersji optymalizacyjnej. Problem komiwojażera, lub jego uogólnienie w postaci problemu chińskiego listonosza to przykład problemu z klasy NP trudnych, o złożoności obliczeniowej O(n!). Polega on na wyznaczeniu jak najlepszej trasy zawierającej w sobie wszystkie punkty (wierzchołki grafu) przechodzącej przez każdy punkt tylko jeden raz oraz zaczynającej i kończącej się w tym samym punkcie.

Rysunek 1 Przykład trasy, która musi odwiedzić wszystkie miasta i być pętlą

2. Metoda

Metoda tutaj wykorzystana to tzw. **Metoda podziału i ograniczeń** zwana inaczej „**Branch And Bound**” Polega ona na rozwijaniu drzewa wyborów (stanów odpowiadających ułożeniu wierzchołków w ścieżce) przy jednoczesnym sprawdzaniu „opłacalności” danego węzła. Zastosowana została tutaj również kolejka priorytetowa, która sprawia, że wybieramy kolejne węzły w kolejności od „najlepiej rokującego”, czyli takiego, o najniższej dolnej granicy.



Rysunek 2 Zobrazowanie rozrostu metody BranchAndBound

2.1 Podstawowe pojęcia:

Dolna granica – liczba będąca ograniczeniem wartości rozwiązania, jakie można uzyskać dzięki rozwinięciu (przejrzeniu potomków) danego węzła

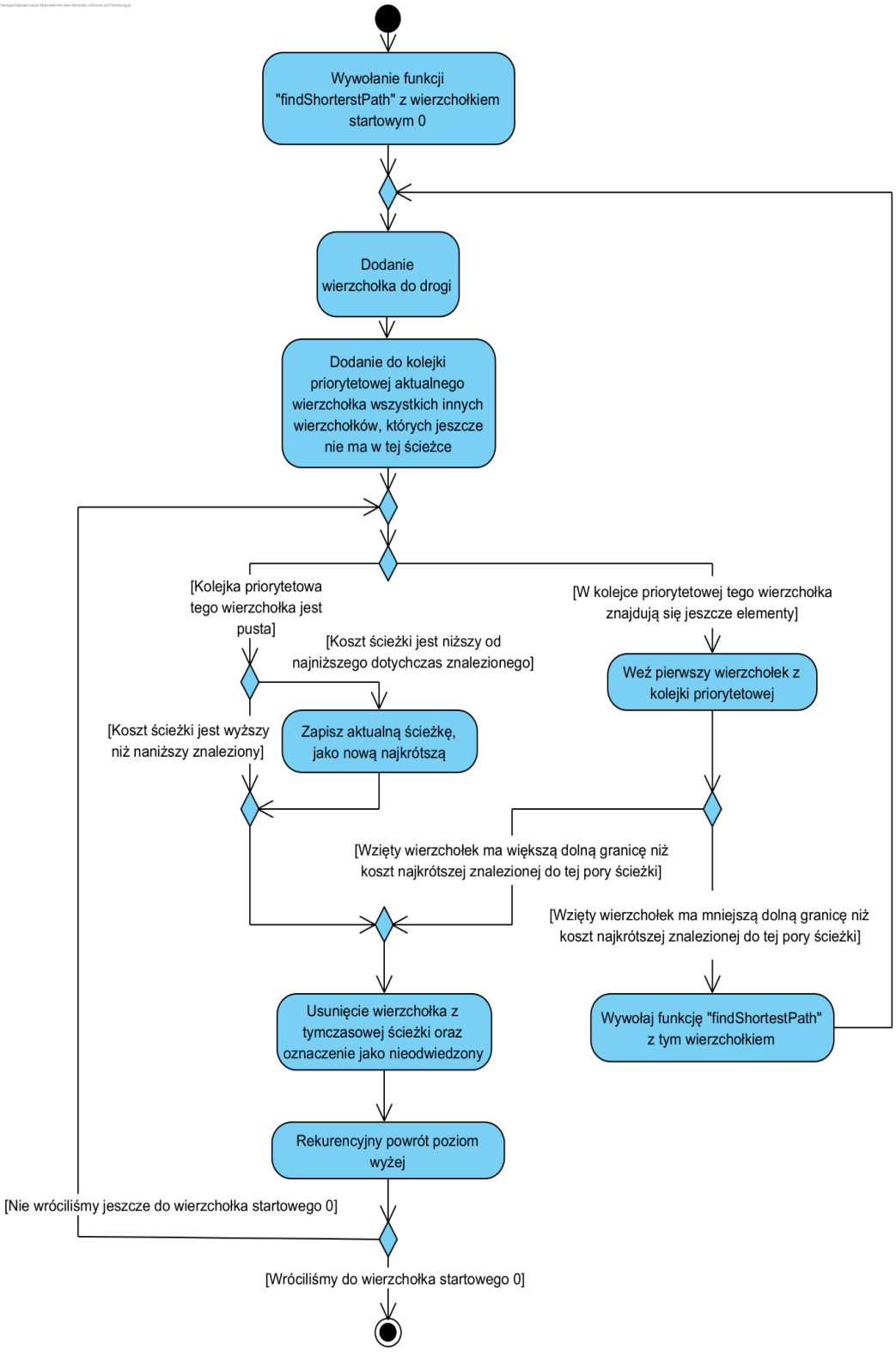
Górna granica – wartość najlepszego dotychczas znalezionego pełnego rozwiązania

Węzeł obiecujący – węzeł, którego dolna granica jest lepsza niż wartość najlepszego znalezionego dotychczas rozwiązania

Węzeł nieobiecujący – węzeł, którego dolna granica jest gorsza od dotychczas znalezionego rozwiązania.

3. Algorytm

Przeszukiwanie tutaj zastosowane to typowy depth search z elementami low cost search, ponieważ dla każdej kolejki wybiera węzeł o najniższym koszcie i to do niego przechodzi.



Rysunek 3 Przebieg algorytmu BranchAndBound

4. Dane testowe

Do sprawdzenia poprawności działania algorytmu oraz do dostrojenia, wybrano następujący zestaw instancji:

- tsp_6_1.txt
- tsp_10.txt
- tsp_12.txt
- tsp_13.txt
- tsp_14.txt
- tsp_15.txt

Dla których dane optymalne zostały zawarte w pliku Optimal.txt, a dodatkowo koszty optymalnej ścieżki zostały wprowadzone do pliku config.txt

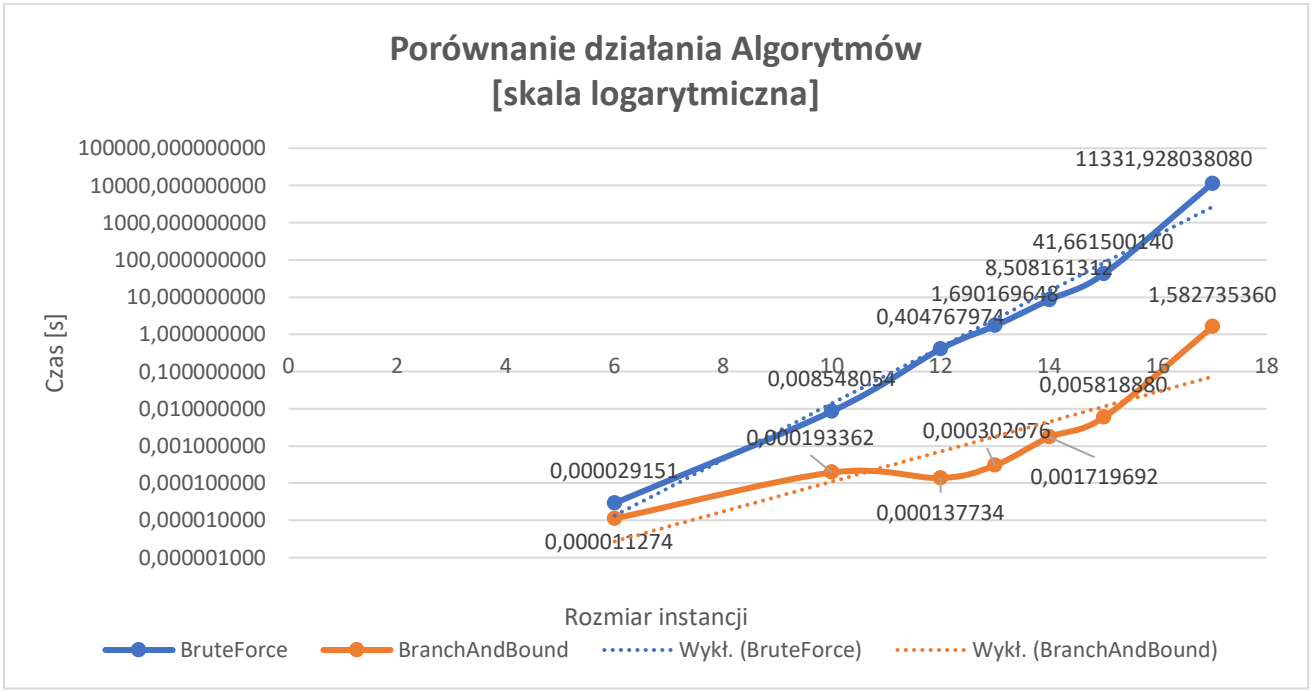
5. Procedura badawcza

Zadanie polegało na zbadaniu zależności czasu rozwiązania problemu od wielkości instancji. W przypadku Branch And Bound nie ma żadnych parametrów do ustawiania, które pozwoliłyby sterować jakością czy szybkością otrzymania rozwiązania. Program jest sterowany plikiem „config.txt", w którym podaje się nazwy plików, na których ma zostać przeprowadzony test, wraz z ilością powtórzeń dla każdego, a także optymalne długości ścieżek dla tych plików.

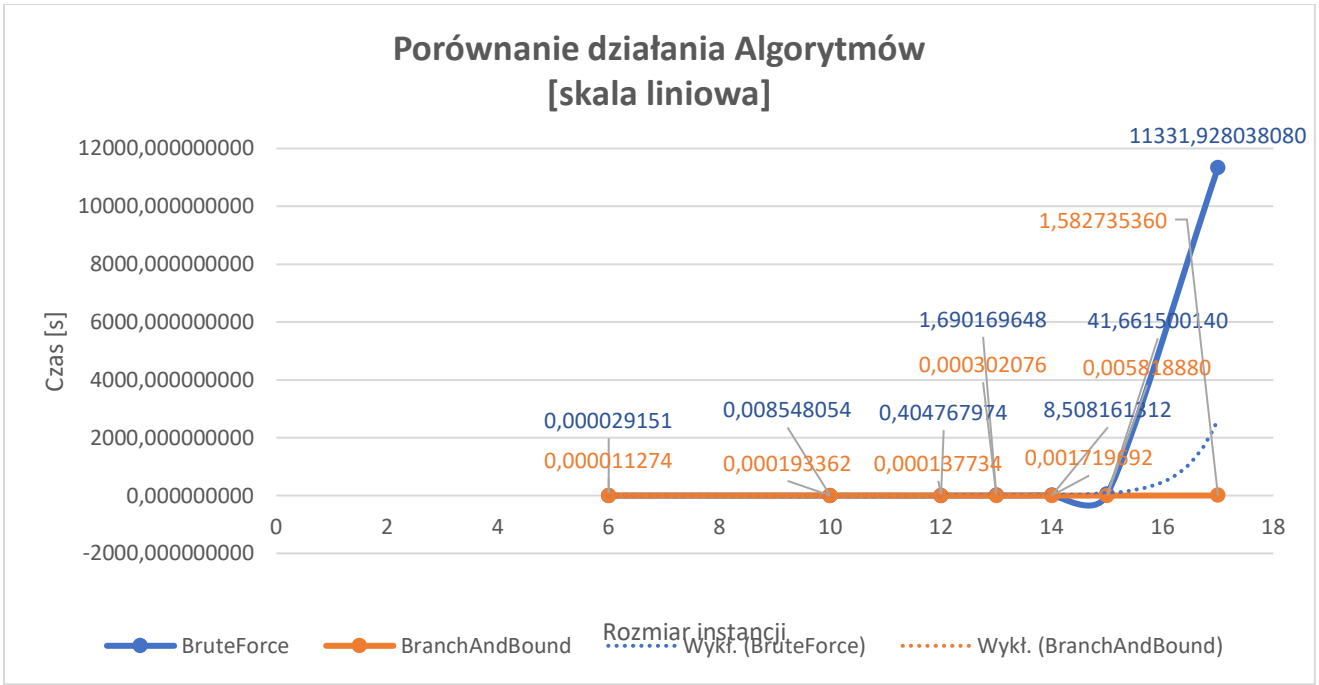
Do pliku wyjściowego „daneWynikowe.txt" wypisywane są czasy wykonania każdego algorytmu, oraz optymalna ścieżka wraz z jej kosztem i tym, o ile rozwiązanie jest gorsze od optymalnego porównawczego. Następnie wyniki zostały obrobione w programie MS Excel.

6. Wyniki

Wyniki zostały zgormadzone w pliku „daneWynikowe.txt”. A następnie zaprezentowane na wykresie (Rysunek 4). Oś Y jest tutaj w skali logarytmicznej, aby było widać wyniki. Dla porównania jeszcze drugi wykres, w którym to Oś Y nie będzie w skali logarytmicznej. Wyniki są od razu przyrównane do tych osiągniętych w poprzednim etapie metodą BruteForce.



Rysunek 4 Wykres zależności czasy wykonania algorytmu od wielkości instancji



Rysunek 5 Wykres taki sam jak na rysunku 4, ale nie w skali logarytmicznej

7. Analiza wyników i wnioski

Przerywaną linią zostały zaznaczone wykładnicze linie trendu, które na obu wykresach zgadzają się z kierunkiem i sposobem podążania wykresu, co potwierdza wstępne założenie o tym, że są to algorytm o złożoności O(n!). Jednak Bardzo wyraźnie można zauważyć (zwłaszcza na wykresie liniowym), że metoda BranchAndBound radzi sobie dużo lepiej od BruteForce'a – jej linia trendu na pierwszym wykresie jest mniej stroma (ma mniejszy współczynnik a), więc granica „opłacalności" wykorzystania tego algorytmu będzie ustawiona dla dużo większych instancji niż dla najprostszej metody BruteForce.