Optymalizacja kombinatoryczna

Karol Bonenberg

26 lutego 2006

1 Treść zadania

Dany jest graf $G = (V, E)$, długości krawędzi w grafie - $l(e) \in Z^+$ dla każdej krawędzie $e \in E$. Zadanie polega na znalezieniu cyklu o największej wartości, czyli znalezienie $p =< v_1, v_2, \ldots, v_m, v_1 >$, takiego, że

$$max_p \left(\sum_{i=1}^{m-1} (l(v_i, v_{i+1}) + l(v_m, v_1) \right). \tag{1}$$

2 Algorytm siłowy

Algorytm siłowy (z ang. brute force, skrótowo BF) jest algorytmem dokładnym, to znaczy, że przeszukuje całą przestrzeń rozwiązań problemu. Dla danego problemu algorytm dokładny polega na znalezieniu wszystkich cykli w grafie, porównaniu ich wartości, a nastepnie wybranie na tej podstawie najlepszego cyklu. Główną ideą algorytmu siłowego jest przeglądanie wszystkich ścieżek zaczynajacych się w danym wierzchołku (o aktualnie najniższym numerze). Niech dany będzie graf $G = (V, E)$ i aktualna ścieżka wygląda następująco $v_0, v_1, \ldots, v_{k-1}, v_k$. Aktualnie dodawany wierzchołek v_{k+1} może być wierzchołkiem:

1. v_0 co oznacza znalezienie cyklu
2. wierzchołkiem nie należącym do cyklu, wtedy następuję dalsze przeglądanie grafu
3. wierzchołkiem będącym już na ścieżce, taki wierzchołek jest ignorowany

Powyższą procedurę można zaimplementować w sposób rekurencyjny. Najpierw dodawany jest wierzchołek do cyklu, dalej następuje rekurencyjne wywołanie funkcji szukającej ścieżki od dodanego wierzchołka, czyli szukanie ścieżki od każdego następnika aktualnego wierzchołka. Posiłkując się powyższym przypadkiem to algorytm w przedstawionym momencie sprawdzając następniki wierzchołka v_k dokona przegląda wszystkich rozwiązania dla każdego następnika v_k. Jeżeli wszystkie następniki wierzchołka v_k zostały już przejrzane to następuje cofnięcie się o jeden krok w celu poszukiwania cyklu poprzez inne następniki, czyli zdjęcie z końca ścieżki wierzchołka v_k i sprawdzenie pozostałych następników wierzchołka v_{k-1}. Algorytm jest tak długo wykonywany dopóki nie zostaną przejrzane wszystkie następniki pierwszego wierzchołka. Następnie usuwany jest pierwszy wierzchołek (dla podanego przykładu v_0) z list następników wszystkich wierzchołków, czyli usuwany jest on z grafu, gdyż zostały podjęte poszukiwania wszystkich cykli zawierających wierzchołek v_0. Ta procedura jest powtarzana tak długo dopóki w grafie nie pozostaną dwa wierzchołki. W celu optymalizacji tego rozwiązania i uniknięcia przeglądania kilka razy tych samych, nie prowadzących do rozwiązań fragmentów ścieżek wprowadzone są dodatkowe oznaczenia. Jeżeli przez dany wierzchołek udało się przeprowadzić chociaż jeden cykl, to ta informacja jest propagowana w rekurencji. Tak wybrane wierzchołki, po przejrzeniu wszystkich ich następników staną się ponownie dostępne w celu znalezienia innego cyklu, który może przez nie przechodzić.

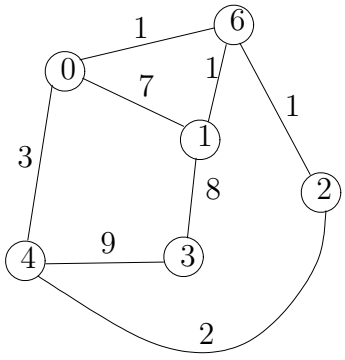
2.1 Złożonośc obliczeniowa algorytmu BF

Ze względu iż problem poszukiwania największego cyklu w grafie w wersji decyzyjnej należy do klasy problemów NP-zupełnych, zatem algorytmy dokładny musi być algorytmem wykładniczym. Ogólnie jego złożoność można wyrazić jako $O(|V| + |E|)(c + 1)$. Gdzie c jest liczbą wygnerowanych cykli. Najwięcej cykli posiada graf pełen. Dla grafu pełnego o n wierzchołkach liczba cykli wynosi:

$$c = \sum_{i=3}^{n} \binom{n}{i} (i-1)!$$

2.2 Przykład działania BF

Na rysunku 1 przedstawiony jest przykładowy graf, na podstawie którego omówione zostanie działanie algorytmów. Algorytm BF rozpocznie swoje działanie od wierzchołka 0, następnie doda jako kolejny wierzchołek 1, następnie 4. W tym momencie dokona z dwóch następników wybierze 0 i znajdzie cykl $< 0, 1, 3, 4, 0 >$ o wartości 27. Cofnie się o jeden krok i zostanie



Rysunek 1: Przykładowy graf dla algorytmów

mu do sprawdzenia kolejny następnik wierzchołka 4, wierzchołek nr 2. I dalej przechodząc graf uzyska cykl $< 0, 1, 3, 4, 2, 5, 0 >$ o długości 28, czyli większy od aktualnego. Kontynując przeszukiwania dojdzie cofnie się do wierzchołka 0 i sprawdzi kolejne jego następniki (5 oraz 4) nie uzyskując cyklu o większej długości. W tym momencie z grafu zostaną usunięte krawędzie łączące wszystkie wierzołki z wierzchołkiem nr 0. I rozpoczęte zostanie poszukiwanie cyklu rozpoczynającego się w wierzchołku 1. I tak dalej, aż do pozastania w grafie wierzchołków 4 i 5, kiedy to z racji nie możności utworzenia cyklu z 2 wierzchołków zostanie zakończone przeszukiwanie grafu i najlepszym cyklem będzie cykl $< 0, 1, 3, 4, 2, 5, 0 >$ o długości 28.

3 Algorytm Branch and Bound

Algorytm typu Branch and Bound (skrótowo BB) oparty jest na algorytmie dokładnym. Wykorzystuje on pewne struktury, które poprzednio służy tylko do nawigacji po grafie w celu szacowania możliwych rozwiązań. Mianowicie, jeżeli z danego wierzchołka zostaną przejrzane wszystkie krawędzie (innymi słowy wszystkie następniki) i żadna z tych decyzji nie przyniesie powodzenia, to dany wierzchołek dodawany jest do tzw. bezużytecznych poprzedników swoich następników. Oznacza to, że krawędzie łączące wierzchołki z bezużytecznymi poprrzednikami nie posłużą do zwiększenia cyklu. Tak zgromadzona informacja wykorzystywana jest podczas oszacowania opłacalności poszukiwań w danym poddrzewie poszukiwań. W trakcie dodawania i odejmowania nieużytecznych wierzchołków liczona jest suma nieużytecznych krawędzi. Jest to dokonane w celu sprawdzenia poniższej zależności: $S_a - S_u > B$ gdzie S_a oznacza sumę wartości wszystkich krawędzi w grafie, S_u sumę krawędzi powiązanych z bezużytecznymi przodkami, B - wartość aktualnie najlepszego cyklu w grafie. Lewa strona tej nierówności tożsama jest sumie wartości ak-

tualnej ścieżki i sumie krawędzi, które nie należą do ścieżki i mogą ją potencjalnie powiększyć. Zatem interpretacja warunku (3) nasuwa się sama. Otóż jeżeli nierówność jest spełniona, to celowe jest poszukiwanie cyklu w danym poddrzewie poszukiwań. Jeżeli nie jest, oznacza to, że nie jest możliwe uzyskanie lepszego cyklu niż aktualnie najlepszy, zatem dokonywane jest odcięcie. Ze względu na czas obliczania sumy wag krawędzi nieużytecznych, operacja ta została wpleciona w algorytm przechodzenia grafu (co zostało wspomniane powyżej). Wybór takiego sposobu odcięcia wymusiło kolejną modyfikację algorytmu. W przypadku, kiedy dalsze poszukiwania zostały odrzucone poprzez odcięcie należało założyć, że taki wierzchołek może prowadzić do cyklu. W przypadku pominięcia tego założenia i dotarciu ponownie do tego wierzchołka z mniejszą wartością krawędzi nieużytecznych odcięcie nie nastąpi i prowadzone będą tam poszukiwania. Jeżeli nie dokonane byłoby takie założenie, a wierzchołek ten prowadziłby do cyklu, to pominięte by mogło zostać rozwiązanie optymalne i algorytm BB nie dawałby optymalnych rezultatów.

3.1 Złożoność obliczeniowa algorytmu BB

Ze względu iż algorytm BB oparty jest na algorytmie BF jego złożoność jest identyczna tzn. wynosi ona $O(|V| + |E|)(c + 1)$. Jednak wartość c wyrażona jest innym wzorem różniącym się jedynie o stałą m

$$c = \sum_{i=3}^{n} \binom{n}{i} (i-1)! - m$$

Gdzie m oznacza liczbę cykli, dla których następuje odcięcie spowodowane zmniejszeniem rozmiaru grafu i przekroczeniem wartości maksymalnego cyklu. Wartość ta w dużej mierze zależy od wag krawędzi stąd trudno ją oszacować.

3.2 Przykład działania BB

Algorytm BB dla przykładu z rysunku 1 będzie działał w początkowej fazie tak samo jak algorytm BF. Poszukując najdłuższego cyklu zaczynajacego się od wierzchołka 0 znajdzie między innymi cykl $< 0, 1, 3, 4, 2, 5, 0 >$ o długości 28. Dopiero w drugiej fazie działania, gdy będzie poszukiwał cyklu zaczynającego się od wierzchołka 1 zajdą odcięcia. Suma wag wszystkich krawędzi w pełnym grafie wynosi 32. Natomiast po usunięci krawędzi do wierzchołka 0, wynosi 21. W związku z tym, że aktualne najlepsze rozwiązanie wynosi 28, zatem dla wierzchołka 1 od razu można zauważyć, że nie będą prowadzone

poszukiwania, ponieważ bez względu na sumę wag krawędzi nieużytecznych nierówność (3) będzie niespełniona. W celu przyspieszenie algorytmu BB, w związku z taką ewentualnością zostało wprowadzone to samo odcięcie, jednak tuż przed uruchomieniem poszukiwania dla grafu bez krawędzi związanych z "pierwszym wierzchołkiem". Wtedy po zaktualizowaniu sumy wag krawędzi w nowym grafie, wystarczy, że najlepszy cykl jest lepszy niż ta suma, to wtedy nie będą dalej usuwane krawędzie z grafu, ani też prowadzone dalsze poszukiwania. Nastąpi zakończenie algorytmu. Liczba odwiedzonych w ten sposób wierzchołków znacznie zmaleje w porównaniu z algorytmem BF. Najlepszy poszukiwany cykl będzie dokładnie tym samym co w przypadku algorytmu BF, a ilość przeszukanych węzłów znacznie spadnie.

4 Algorytm z heurystyką

Algorytm heurystyczny oparty jest na zupełnie innej idei niż dwa wyżej wspomniane. Do konstrukcji rozwiązania wykorzystuje się zmodyfikowany algorytm Prima poszukiwania minimalnego drzewa rozpinającego. Modyfikacja polega na poszukiwania maksymalnego drzewa rozpinającego. Zamiast wybierać najmniejszą krawędź wychodząca na zewnątrz aktualnego drzewa, wybiera tę z największą wartością. Po utworzeniu takiego drzewa rozpinającego przez dodanie pojedynczej krawędzi uzyskuję się cykl o stosunkowo dużej wartości. Krawędź, którą dołącza się jest ta z pozostających poza drzewem rozpinającym, która ma największą wartość.

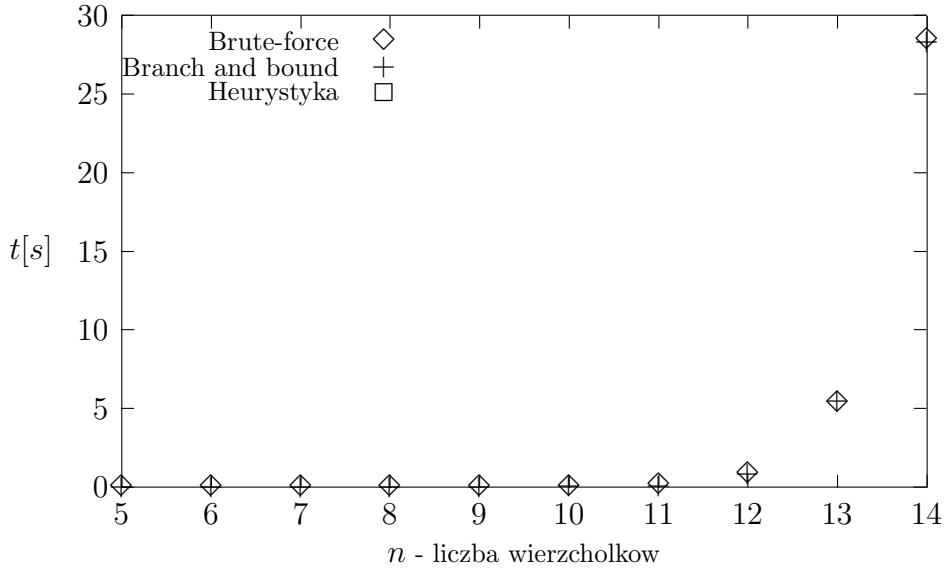
4.1 Złożoność obliczeniowa heurystyki

Na całkowitą złożoność obliczeniową heurystyki składają się suma złożoności dwóch algorytmów. Wykorzystany do tworzenia drzewa rozpinającego algorytm Prima działa w czasie wielomianowym tj. ma złożoność $O(|E|log|V|)$,

4.2 Przykład działania heurystyki

5 Eksperymenty obliczeniowe

W celu zbadania jakości algorytmów zostały przeprowadzone różnego rodzaju testy wydajności. Pierwszym kryterium porównawczym były czasy działania algorytmów w funkcji rozmiaru instancji tj. liczby wierzchołków. Wyniki tego eksperymentu znajdują się na wykresie 2 oraz zostały zestawione w tabeli XXX. Wyraźnie na nich widoczna jest wykładnicza zależność czasu wykonywania dla algorytmów dokładnych (BB/BF). Kolejnym kryterium pod



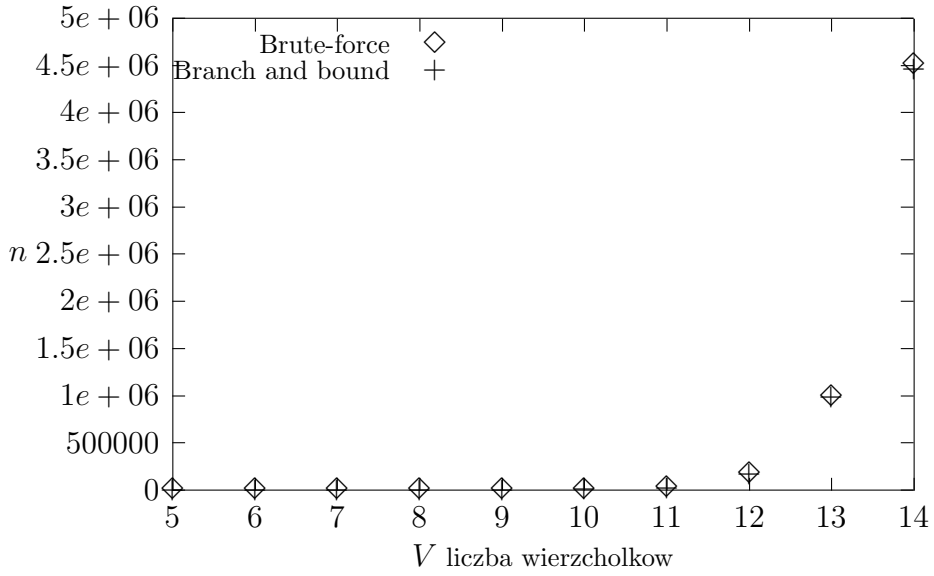
Rysunek 2: Czas działania algorytmów w funkcji rozmiaru instancji

jakim były sprawdzane algorytmy była liczba odwiedzanych wierzchołków oraz liczba znalezionych cykli. Testy te dotyczyły algorytmów BB i BF. Wykonane były w celu sprawdzenia korzyści płynących z odcięć w algorytmie BB. Wyniki tych testów przedstawione są na wykresiach 3 i XXX, a zestawie wartości liczbowych znajduje się w tabelach XXX i XXX.

6 Porównanie algorytmów

7 Wnioski

Ze względu na złożoność samego problemu algorytm dokładny (BF) i jego modyfikacja w postaci odcięć (BB) działają w czasie wielomianowym. Czas działania tych algorytmów mniej więcej się pokrywają, gdyż to, co algorytm BB uzyskuje na odcięciach jest tracone przez dodatkowe obliczenia jakie musi wykonać w trakcie przeglądu grafu - tworząc sumę nieużytecznych krawędzi. Co jest warte zauważenia przejrzenie mniejszej ilości rozwiązań, nie obarczone dodatkowymi warunkami może okazać się znacznie szybsze, gdyż sprowadza się do stosunkowo prostych operacji na wektorach i listach, podczas prac nad algorytmem BB uzyskiwał on dość często rozwiązania w czasie dłuższym, a wynikało to z obliczania wartości krawędzi nieużywanych. Przeniesienie



Rysunek 3: Porównanie liczby odwiedzanych wierzchołków - n dla BF i BB

tego ciężaru na czas przeglądania grafu i wpleciennie w naturalnie wykonywane operacje sprawiło, że algorytm BB uzyskiwał lepsze wyniki niż algorytm BF. Skuteczność algorytmu BB spada wraz ze zwiększeniem się liczby krawędzi, gdyż liczba potencjalnych krawędzi nieużytecznych rośnie, a co za tym idzie wzrasta liczba koniecznych operacji dodawnia i odejmowania. Ze względu na konstrukcję problemu można również wnioskować, że może być ciężkie uzyskanie szacowania w krótszym czasie - możliwe, że stworzenie własnych struktur danych do reprezentacji grafu przyspieszyłoby działanie, jednak w implementacji ograniczono się do wykorzystania biblioteki STL oferowanej przez język C++. Z kolei algorytm heurystyczny swoją konstrukcją odbiega od wcześniejszych - zmniejszenie liczby przeglądanych węzłów uzyskuje przez zastosowanie zmodyfikowanego algorytmu Kruskala, co sprawia, że czas jego działania jest mniejszy niż algorytmów dokładnych, a jakość rozwiazań jest zadowalająca. Zatem podczas wyboru algorytmu należy podjąć decyzję, czy zależy nam na czasie, czy też na jakości rozwiązań. Jego niewątpliwą zaletą jest fakt uzyskiwania rozwiązań dla grafów dużych. Szczególnie jest to zauważalne dla grafów pełnych o rozmiarach większych niż 20. Dla takich instancji problemu, algorytmy dokładne działają dość długo.