

Bounded Diameter Minimum Spanning Tree

Teoria Obliczeń i Złożoność Obliczeniowa

Adrian Mucha

9 stycznia 2021

[1]

1 Wprowadzenie

Mając ważony, nieskierowany graf G i dodatnią liczbę D w problemie *Bounded-Diameter Minimum Spanning Tree (BDMST)* szukamy najniższego kosztem drzewa rozpinającego spośród wszystkich drzew rozpinających G , których ścieżki składają się z co najwyżej D krawędzi. Formalnie *BDMST* jest drzewem $T \subset E$ na $G = (V, E)$, którego średnica jest nie większa niż D . *BDMST* ma na celu znalezienia drzewa rozpinającego o minimalnym koszcie $w(T) = \sum_{e \in T} w(e)$. Zauważając do grafów Euklidesowych, czyli takich w których wierzchołki są punktami na przestrzeni Euklidesowej a wagi krawędzi reprezentują dystans między parami wierzchołków nazywamy *Euclidean BDMST*.

Problem BDMST jest NP-trudny dla $4 \leq D < |V| - 1$, oraz trudny w aproksymacji co motywuje w poszukiwaniach efektywnej strategii opartej na heurystykach, które potrafią szybko znaleźć BDST o niskim koszcie.

Oczywistym zastosowaniem *EBDMST* jest znalezienie najtańszej sieci kabli lub rur by połączyć zbiór miejsc zakładając, że koszt połączenia zależy od jego długości.

2 Podstawowe heurystyki

Poniżej przedstawiono przegląd podstawowych heurystyk

2.1 One-Time Tree Construction (OTTC)

Jest to zachłanny algorytm oparty na heurystyce która oblicza średnicę drzewa rozpinającego w każdym kroku i upewnia się, że następny wierzchołek nie przekroczy ograniczenia. A następnie do budowanego drzewa rozpinającego w każdym kroku dodaje krawędź o najniższym koszcie, który nie łamie obostrzeń na średnicę. Dodanie wierzchołka wymaga pracy $\mathcal{O}(n^2)$, a całość wykonywana jest $n - 1$ razy, więc całkowity czas pracy algorytmu rozpoczynającego w pojedynczym wierzchołku to $\mathcal{O}(n^3)$. Aby znaleźć najniższe kosztem BDST, algorytm

OTTC uruchamiany jest dla każdego wierzchołka grafu (n razy), więc całkowita złożoność wynosi $\mathcal{O}(n^4)$ [2].

2.2 Center-Based Tree Construction (CBTC)

W drzewie o średnicy D , żaden wierzchołek nie znajduje się dalej niż $\frac{D}{2}$ skoków (lub krawędzi) od korzenia. Dzięki temu zabiegowi otrzymujemy szybszy algorytm bazujący na algorytmie Prima, który poprawia OTTC dzięki budowaniu BDST od środka drzewa. Zapamiętywanie stopnia wierzchołków i zapewnianie, że żaden nie przekroczy głębokości $\lfloor \frac{D}{2} \rfloor$ pozwala zaoszczędzić ciągłego przeliczania średnicy drzewa przed dołączeniem wierzchołka do BDST. Otrzymujemy nieco lepszą złożoność $\mathcal{O}(n^3)$ w porównaniu do OTTC (tutaj również należy rozważyć algorytm startując z każdego wierzchołka osobno i wybrać drzewo o najniższym koszcie) [3].

2.3 Randomized Tree Construction (RTC)

W losowej konstrukcji drzewa korzeń (centrum) jest wybierany na początku jako losowy wierzchołek (jeśli D jest parzyste) lub losowane są dwa wierzchołki połączone ze sobą (jeśli D jest nieparzyste). Każdy następny dołączany wierzchołek jest również wybierany losowo w sposób zachłanny, taki że przyłączenie wierzchołka nie przekroczy ograniczenia D na średnicę drzewa. Jest to identyczny w implementacji algorytm jak CBTC z tą różnicą że wprowadzono element losowości przy wybieraniu wierzchołków. Również posiada tę samą złożoność $\mathcal{O}(n^3)$.

2.4 Pozostałe heurystyki

Po skonstruowaniu BDST jednym z algorytmów (CBTC lub RTC) dodatkowo sprawdzane jest dla każdego wierzchołka $v \in V$ którego głębokość jest większa niż 1 czy można go odłączyć i połączyć z innym wierzchołkiem BDST mającym niższy stopień głębokości krawędzią o mniejszym koszcie.

Heurystyka przetrzymuje posortowaną macierz kosztów w celu szukania krawędzi o niskim koszcie by dodać do BDST wierzchołek v . Aby zachować kolejność, używa się macierzy pomocniczej pamiętającej indeksy [4].

3 Heurystyki CBLSoC-lite oraz CBLSoC

Center-Based Least Sum of Costs *Lite* wybiera wierzchołek na korzeń (lub dwa wierzchołki w przypadku parzystej średnicy) z najmniejszym kosztem do reszty wierzchołków $v_0 = \arg \min_v \left\{ v \in V \mid \sum_{u \in N_G(v)} w((v, u)) \right\}$. Podążając tym podejściem, kolejne wierzchołki również są wybierane na podstawie najniższej sumy kosztów do reszty wierzchołków z uwzględnieniem heurystyki CBTC, a zatem upewnianiem się, że żaden wierzchołek nie przekroczy głębokości $\lfloor \frac{D}{2} \rfloor$. Otrzymany algorytm (wersja *Lite*) wykonuje pracę $\mathcal{O}(n^2)$.

CBLSoc jest wariacją pierwszego, gdyż różni się tym, że wybór korzenia nie odbywa się na podstawie najniższego kosztu lecz powtarza się algorytm *Lite* n razy dla każdego wierzchołka grafu. Całkowity czas działania algorytmu wynosi więc $\mathcal{O}(n^3)$.

Podobnie jak w CBTC i RTC otrzymujemy jednakowe złożoności natomiast wynikowe drzewa rozpinające posiadają niższe koszty.

4 Quadrant Centers-based (QCH)

Literatura

- [1] C. Patvardhan, V. Prem Prakash, and Anand Srivastav. Fast heuristics for large instances of the euclidean bounded diameter minimum spanning tree problem. *Informatica (Slovenia)*, 39(3), 2015.
- [2] Narsingh Deo and Ayman M. Abdalla. Computing a diameter-constrained minimum spanning tree in parallel. In Gian Carlo Bongiovanni, Giorgio Gambosi, and Rossella Petreschi, editors, *Algorithms and Complexity, 4th Italian Conference, CIAC 2000, Rome, Italy, March 2000, Proceedings*, volume 1767 of *Lecture Notes in Computer Science*, pages 17–31. Springer, 2000.
- [3] Bryant A. Julstrom. Greedy heuristics for the bounded diameter minimum spanning tree problem. *ACM J. Exp. Algorithmics*, 14, 2009.
- [4] Alok Singh and Ashok Kumar Gupta. Improved heuristics for the bounded-diameter minimum spanning tree problem. *Soft Comput.*, 11(10):911–921, 2007.