Problem k-minimalnego drzewa rozpinającego

Gabriel Budziński
254609

May 27, 2023

1 Wprowadzenie

Weźmy graf nieskierowany $G = (V, E)$ o n wierzchołkach $w \in V$, nieujemnych kosztach c_e krawędzi $e \in E$ oraz liczbę $k \in \mathbb{N}$. Problem k-minimalnego drzewa rozpinającego (*ang.* kMST - k-minimal spanning tree, MSkT - minimal spanning k-tree) polega na poszukiwaniu drzewa w G o minimalnym koszcie, w które wchodzi co najmniej k wierzchołków G. Problem ten jest NP-trudny nawet dla V należących do płaszczyzny Euklidejskiej. Problem ten jest silnie związany z innym, występującym we wcześniejszych latach w literaturze [1] - minimum weight k-cardinality tree, którego rozwiązaniem jest znalezienie w grafie G poddrzewa o k krawędziach.

2 k-cardinality tree

2.1 Opis problemu

Weźmy graf $G = (V, E)$ ze zbiorem wierzchołków V i krawędzi E. Moce zbiorów V i E to odpowiednio $n = |V|$ oraz $m = |E|$. Dla każdej krawędzi $e \in E$ dana jest waga $w(e) \in \mathbb{R}$, a waga zbioru $E' \subseteq E$ jest definiowana jako $\sum_{e \in E'} w(e)$.

Drzewem w G jest podgraf $T = (V(T), E(T))$ taki, że T nie zawiera cykli i jest spójny. Będziemy używać notacji $w(T)$ opisując $w(E(T))$. Moc $|T|$ zbioru T jest mocą $E(T)$. Dla zadanego k, gdzie $1 \leqslant k \leqslant n - 1$ k-cardinality tree jest drzewem T o mocy $|T| = k$. Jeśli $k = n - 1$ to T jest drzewem rozpinającym G. Zadane jest znalezienie takiego T, że $w(T) = \min_{T' \subseteq G} w(T')$. Dla $k = n - 1$ takim T jest minimalne drzewo rozpinające, które można znaleźć w czasie wielomianowym algorytmem zachłannym (Kruskal [2], Prim [3]). Dla ustalonego k problem jest różnież rozwiązywalny przez wyliczenie wszystkich możliwych drzew. Jeśli mamy wagi zadane są dla wierzchołków, to możemy rozpatrywać graf krawędziowy do zadanego grafu G.

2.2 Zastosowania w praktyce

Powyższy problem pojawia się w najmie pól naftowych [4]. Rząd ma następującą regułę "50%" obejmującą morskie pola naftowe: jeśli firma najęła pole naftowe ma ona ustaloną liczbę lat, dajmy na to 5, aby eksploatować to pole. Po upływie tego czasu firma ma obowiązek zwrócić co najmniej 50% najętego pola. Ponadto, oddawana część pola musi być spójna. Oczywistym celem z punktu widzenia firmy jest zwrot częsci o najmniejszej wartości (i zachowanie części o wartości największej). W pracy [4] pola naftowe mają postać prostokąta podzielonego na mniejsze kwadraty. Firma, która najmuje pole ma 5 lat na zebranie informacji o wartości w_i każdego z podkwadratów. Część pola, którą firma odda odpowiada podzbiorowi co najmniej 50% podkwadratów, który jest spójny i ma najmniejszą całkowitą wartość wszystkich w_i. Aby zamodelować spójność weźmy graf liniowy do oczekiwanego, który jest grafem kratowym (patrz 1). Spójny podzbiór kwadratów w prostokącie odpowiada spójnemu podgrafowi G. Ponieważ wagi kwadratów odpowiadają wagom wierzchołków G, to rozwiązując problem k-CARD TREE w grafie wierzchołkowym do G, gdzie $k \geqslant \frac{n}{2}$ mamy optymalną część pola do zwrotu.

1	2	3	4	5
6	7	8	9	10
11	12	13	14	15

Table 1: Pole naftowe

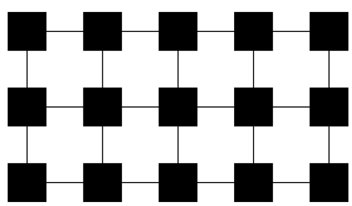

Figure 1: Graf G

2.3 Złożoność problemu k-CARD TREE

Pokażemy, że problem drzewa Steinera, który jest powszechnie znanym z bycia silnie NP-trudnym [5], może być zredukowany do k-CARD TREE. W uprzednim problemie zadany jest podzbiór $S \subseteq V$ i próbujemy znaleźć drzewo Steinera o minimalnej wadze, czyli drzewo T w G takie, że $S \subseteq V(T)$ oraz $w(T)$ jest najmniejsza. Dowód redukcji znajduje się w pracy [1].

2.4 Najnowsze metody poszukiwania rozwiązań

2.4.1 Przeszukiwanie sąsiedztwa

W pracy [6] podano wariant metody *Variable neighbourhood search* (VNS [7]) przygotowany do problemu k-CARD TREE. Przestrzeń rozwiązań $\mathcal{S}_k$, będąca zbiorem wszystkich podrzew T grafu G o dokładnie k krawędziach. Moc tej przestrzeni to $|\mathcal{S}_k| = \binom{n}{k+1} \cdot O(k^k)$. Odległość między dwoma drzewami T_1 oraz T_2 zdefiniowano jako:

- $\rho(T_1, T_2) = |E_{T_1} \backslash E_{T_2}| = |E_{T_2} \backslash E_{T_1}|$
- $\eta(T_1, T_2) = |V_{T_1} \backslash V_{T_2}| = |V_{T_2} \backslash V_{T_1}|$

Wprowadzona przez Uroševića, Brimberga i Mladenovića metoda *Variable neighbourhood decomposition search* (VDNS [8]) rozszerza VNS na dwupoziomowy VNS po dekompozycji problemu.

Powyższa metoda porównana została ze standardowym VNS oraz dwoma implementacjami tabu search (TS-1 [9], TS-2 [10]). Testy przeprowadzone zostały dla $n = |V|$ z przedziału [500,5000]. Na podstawie eksperymentów wykazano lepsze od konkurencji wyniki algorytmu VNDS, zarówno w postaci rozwiązań bliżej optimum, jak i krótszego czasu wykonania.

2.4.2 Branch-and-bound

W pracy [11] rozszerzającej podejścia *branch-and-bound* i *branch-and-cut* odpowiednio z prac [12] oraz [13] rozważany jest ukorzeniona wersja problemu k-CARD TREE, zwana dalej RKCTP, gdzie zadany jest wierzchołek $r \in V$, a rozwiązanie musi zawierać r.

Celem zapronowanej w pracy metody jest rozwiązanie k-CARD TREE (zwanego tu KCTP) jako sekwencji $n-k+1$ nierosnących podproblemów RKCTP. Metoda bazowana jest na następującym pomyśle: mając zadany wierzchołek $v \in V$, weźmy $G' = G - v$. KCTP w G (którego rozwiązanie przedstawmy jako KCTP(G)) zawiera wierzchołek v lub nie. Jeśli KCTP(G) zawiera v, to problem zredukowany jest od RKCTP zakorzenionego w v (którego optymalne rozwiązanie przedstawmy jako RKCTP(G, v)). W przeciwnym przypadku v nie należy do optymalnego rozwiązania KCTP w G i może być odrzucony. Ta obserwacja prowadzi do następującego wniosku:

$$KCTP(G) = min\{RKCTP(G, v), KCTP(G')\}$$

Problem KCTP został więc sprowadzony do sekwencji podproblemów RKCTP, które będą rozwiązywane metodą *branch-and-bound*. Waskie ograniczenia na optymalne rozwiązanie RKCTP z pracy [14] oraz prosta i wydajna strategia rozgałęzień są kluczowe do minimalizacji kosztów przeszukiwania. Aby rozwiązać RKCTP metodą *branch-and-bound* definiujemy podproblem P na zbiorach krawędzi (S_1, S_0). Zbiór S_1 jest złożony z krawędzi drzewa zakorzenionego w r, zwanych *ustalonymi krawędziami*. Zbiór S_0 jest zbiorem *zakazanych krawędzi*. S_1 jest rozwiązaniem P jeśli ma dokładnie k krawędzi i nie ma żadnych krawędzi z S_0. W podproblemie P, jeśli S_1 jest rozwiązaniem to P jest podproblemem terminalnym w drzewie przeszukiwań. W przeciwnym przypadku, jeśli $E \backslash (S_1 \cup S_0)$ jest niepusty to wybieramy krawędź e z tego zbioru taką, że $S_1 \cup \{e\}$ też jest drzewem oraz definiujemy dwa nowe podproblemy: krawędź e dodajemy do zbiorów S_1 albo S_0. Poprzez te dwie operacje podczas kroku rozgałęziania tworzone są dwa nowe podproblemy RKCTP takie, że optymalne osiągalne rozwiązanie P istnieje tylko w jednej z gałęzi. W tej metodzie rozważaną krawędzią e jest krawędź z $E \backslash (S_1 \cup S_0)$ o minimalnej wadze. Ponadto strategią wyboru gałęzi jest *depth-first-search*.

Podczas testów metodę porównano z dwoma innymi dającymi rozwiązanie KCTP lub jego ograniczenie dolne: k pierwszych kroków algorytmu Kruskala [2] oraz ogranicznie Kataoka'i [14]. Zaproponowana w pracy metoda działała lepiej od obydwu wymienionych, częściej zwracając rozwiązanie optymalne oraz dając lepsze ograniczenia dolne.

3 *k*-spanning tree

3.1 Heurystyki

W pracy [15] rozpatrzono trzy podejścia heurystyczne do rozwiązywania problemu MSkT.

3.1.1 Heurystyki oparte na podejściu *Greedy*

- **Algorytm GreedyA**
  Algorytm oprąty jest na idei algorytmu Prima [3]. Na początku wybieramy krawędź o najmniejszej wadze w G. Następnie konstruujemy klikę startową W o rozmiarze $k+1$ poprzez dodawanie do W wierzchołka m^* o minimalnej sumie wag $\sum_{j \in V(W)} w(m^*, j)$ wraz z tymi krawędziami. Po otrzymaniu kliki wykonujemy algortym z pracy [16], który domyślnie startował z losowej kliki.

3.1.2 Heurystyki oparte o progrmowanie dynamiczne

- **Algorytm DPA**
  Algorytm składa się $|V| + 1$ kroków, przez pierwsze k kroków rozbudowujemy drzewo poprzez dodawanie kolejnych wierzchołków minimalizując sumę wag krawędzi do niego prowadzących z aktualnego drzewa. W pozostałych krokach wyznaczamy k-klikę minimalną pod względem sumy wag krawędzi do każdego z wierzchołków. W $|V| + 1$ kroku wyznaczamy znalezione drzewo.

3.1.3 Heurystyki oparte o iteratywną poprawę rozwiązania wyjściowego

- **Procedura CONNECT**
  Procedura łącząca dwie kliki o rozmiarze $\leqslant k$ minimalizując przy tym wagę dodawanych krawędzi.

- **Procedura COMPLEMENT**
  Procedura tworząca k-drzewo z częściowego k-drzewa, przez które rozumiemy graf posiadający wszystkie wierzchołki oraz podzbiór krawędzi k-drzewa. Klu procedury jest dodawanie krawędzi do częściowego k-drzewa minimalizując przy tym wagę tych krawędzi tak, aby konstruowany graf był k-drzewem.

- **Algorytm RA**
  Algorytm polega na rekursywnym konstruowaniu nowych rozwiązań z rozwiązania wyjściowego na podstawie klik zawierających się w tym rozwiązaniu przy pomocy procedur COMPLEMENT oraz CONNECT.

- **Algorytm FRA**
  Algorytm rozszerzający RA, w którym przy każdym rekursywnym wywołaniu jedynie n klik o najmniejszej wadze jest branych pod uwagę przy kolejnych krokach algorytmu co przyśpiesza poszukiwania optymalnego drzewa.

Na podstawie badań czasu wykonania oraz jakości znalezionego rozwiązano wykazano, że algorytm RA startujący z rozwiązania obliczonego przy pomocy DPA okazał się bardzo kompetatywny ze względu na szybki czas wykonania dla małego k oraz znalezienie optymalnego k-drzewa rozpinającego dla wszystkich wygenerowanych instancji problemu. Dla znacznych k plecono stosowanie algortym FRA bazujący na rozwiązaniu generowanym przez GreedyA ze względu na niewielki wzrost czasu obliczeń wraz ze wzrostem k w porównianiu do RA-DPA.

3.2 Algorytm hybrydowy: tabu search i kolonia mrówek

W pracy [17] wypracowano ulepszenie algorytmu *tabu search* (TS) dla problemu k-drzewa rozpinającego. TS z pracy [17] wykazuje się skutecznym znajdywaniem dobrego rozwiązania w relatywnie małej przestrzeni lokalnej, jednakże możliwości eksploracji poza tę przestrzeń są niewielkie. Tę własność posiada natomiast algortym *ant colony optimisation* (ACO) opisany w pracy [18], co wywnioskowano po znakomitych wynikach dla małych k. Zaproponowany algorytm prezentuje się następująco:

- *Krok 1 (Generowanie rozwiązania wyjściowego).* Rozpoczynając od losowego wierzchołka stosujemy algorytm Prima [3] aż uzyskamy k-poddrzewo.

- *Krok 2 (Inicjalizacja parametrów).* Inicjalizujemy listę *tabu*, jej czas pamiętania tl_{len} oraz kryteria aspiracji.

- *Krok 3 (Lokalne przeszukiwanie tabu-search).* Przeszukujemy sąsiedztwo używając TS i zapisujemy rozwiązania minimalne. Jeśli aktualne tl_{len} przekroczy tt_{max} przechodzimy do stanu 4. W przeciwnym przypadku powracamy do stanu 2.

- *Stan 4 (Procedura dywersyfikacyjna oparta na kolonii mrówek).* Rozszerzamy obszar przeszukiwań na podstawie ACO aby zwiększyć różnorodność rozwiązań.

- *Stan 5 (Ostateczny).* Jeśli przekroczyliśmy dozwolony czas obliczeniowy, kończymy algorytm i zwracamy rozwiązanie. W przeciwnym przypadku powracamy do stanu 2.

Testy numeryczne opisane w pracy wykazały lepsze zachowanie algorytmu od algorytmów, na których bazował.

3.3 *k*-spanning tree na okręgach

Na koniec przyjrzyjmy się pracy [19], w której rozpatrywany jest szczególny przypadek MSkT, w którym wierzchołki grafu leżą na okręgu. Uprzednio R. Ravi [20] pokazał między innymi, że jeśli wierzchołki grafu leżą na krawędzi figury wypukłej, a wagi krawędzi odpowiadają euklidejskim odległościom, to MSkT można rozwiązać w czasie wielomianowym stosując programowanie dynamiczne. Zauważono, że krawędzie każdego minimalnego drzewa rozpinającego o k wierzchołkach nie mogą się wzajemnie przecinać, a przez to rozwiązanie optymalne MSkT dla wielokąta może być skonstruowane z rozwiązń pomniejszych wielokątów otrzymanych przez triangulację wielokąta wyjściowego. W kolejnej części zaproponowany jest algorytm dla okręgu, pod warunkiem, że żadna para wierzchołków nie leży na tej samej średnicy. Knecht i Jungnickel budując na pomysłach Ravi'ego przedstawiają bardziej ogólny algorytm, który nie wymaga powyższego założnia i ma złożoność obliczeniową rzędu $O(n^2k)$. Klu algorytmu jest poszukiwanie najkrótszej ścieżki zawierającej k wierzchołków, która nie zmienia kierunku po okręgu. Poprawność tego spostrzeżenia podpierją dwa lematy:

- Żaden z wierzchołków MSkT nie ma stopnia większego niż 2.
- Najkrótsza ścieżka o k wierzchołkach z v_{i_1} do v_{i_k} wraz z krawędzią $v_{i_1}v_{i_k}$ tworzą k-kąt.

4 Podumowanie

Wyżej cytowane prace składają się na aktualny stan wiedzy o problemie k-drzewa rozpinającego.

References

[1] M. Fischetti, H. W. Hamacher, K. Jornsten, and F. Maffioli, *Weighted k-cardinality trees: Complexity and polyhedral structure*. PhD thesis, Universität Kaiserslautern, 1992.

[2] J. B. Kruskal, "On the shortest spanning subtree of a graph and the traveling salesman problem," *Proceedings of the American Mathematical Society*, vol. 7, 1956.

[3] R. Prim, "Shortest connection networks and some generalizations," *Bell System Technical Journal*, vol. 36, 1957.

[4] H. W. Hamacher and K. Jörnsten, "Optimal relinquishment according to the norwegian petroleum law: A combinatorial optimization approach," *Energy, Natural Resources and Environmental Economics*, 1993.

[5] M. R. Garey and D. S. Johnson, *Computers and Intractability. A Guide to the Theory of NP-Completeness*. Freeman and Company, 1979.

[6] D. Urošević, J. Brimberg, and N. Mladenović, "Variable neighborhood decomposition search for the edge weighted k-cardinality tree problem," *Computers and operational research*, vol. 31, 2004.

[7] N. Mladenović and P. Hansen, "Variable neighborhood search," *Computers and operational research*, vol. 24, 1997.

[8] P.Hansen, N. Mladenović, and D. Perez-Britos, "Variable neighborhood decomposition search," *Journal of Heuristics*, 2001.

[9] N. Mladenović and D. Urošević, "Variable neighborhood search for the k-cardinality tree," *Metaheuristics: Computer Decision-Making*, 2001.

[10] J. Kurt and L. Arne, "Tabu search for weighted k-carfinality trees," *Asia-Pacific Journal of Operational Research*, vol. 14, 1997.

[11] L. Simonetti, F.Protti, Y. Frota, and C. de Souza, "New branch-and-bound algorithms for k-cardinality tree problems," *Electronic notes in descrete mathematics*, vol. 37, 2011.

[12] F. Quintao, A. Cunha, and G. Mateus, "Integer programming formulations for the k-cardinality tree problem," *Electronic Notes in Discrete Mathematics*, vol. 30, 2008.

[13] M. Chimani, M. Kandyba, and I. L. nad P. Mutzel, "Obtaining optimal k-cardinality trees fast," *ACM Journal of Experimental Algorithmics*, vol. 14, 2009.

[14] S. Kataoka, N. Araki, and T. Yamada, "Upper and lower bounding procedures for the minimum rooted k-subtree problem," *European Journal of Operational Research*, vol. 122, no. 3, 2000.

[15] R. E. Shangin and P. M. Pardalos, "Heuristics for minimum spanning k-tree problem," *Procedia computer science*, vol. 31, 2014.

[16] H. Beck and A. Candia, "Heuristics for minimum spanning k-trees," *Investigation Operativa*, vol. 9, 2000.

[17] H. Katagiri, T. Hayashida, I. Nishizaki, and Q. Guo, "A hybrid algorithm based on tabu search and ant colony optimization for k-minimum spanning tree problems," *Expert systems with applications*, vol. 39, 2012.

[18] C. Blum and M. J. Blesa, "New metaheuristic approaches for the edge-weighted k-cardinality tree problem," *Computers and operations research*, vol. 32, 2005.

[19] T. Knecht and D. Jungnickel, "A note on the k-minimum spanning tree problem on circles," *Operations research letters*, vol. 44, 2016.

[20] R. Ravi, R. Sundaram, M. V. Marathe, D. J. Rosenkrantz, and S. S. Ravi, "Spanning trees—short or small," *SIAM Journal on Discrete Mathematics*, vol. 9, 1996.