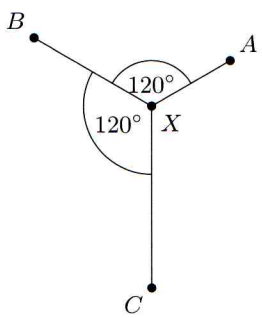
Drzewo Steinera: jedno zagadnienie, mnóstwo problemów

Marek CYGAN[*], *Marcin PILIPCZUK*[*]

W naszym zespole badawczym analizujemy wiele różnych zagadnień, które występują w różnych aspektach. Nic w tym dziwnego, wszak algorytmika jest bardzo szeroką dziedziną. Zdarza się i tak, że to samo zagadnienie pojawia się w wielu kontekstach. Taka sytuacja ma miejsce w przypadku problemu drzewa Steinera.

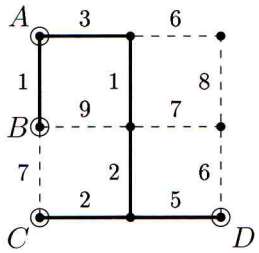
Zanim przejdziemy do formalnej definicji, przyjrzyjmy się nastepującemu klasycznemu problemowi geometrycznemu. Książę Piotruś jest władcą trzech miast: A, B i C. Chce je połączyć drogami tak, by z każdego miasta dało się dojechać do każdego innego, i oczywiście pragnie zminimalizować koszt budowy, czyli sumę długości zbudowanych dróg. Jedną z dostępnych opcji jest zbudowanie dróg wzdłuż dwóch krótszych boków trójkąta ABC. Jednak, co dla niektórych Czytelników może być zaskakujące, można lepiej! O ile tylko każdy kąt wewnętrzny trójkąta ABC ma miarę mniejszą niż $120°$, taniej jest zrobić tak: znajdujemy wewnątrz trójkąta ABC punkt X, taki że $\sphericalangle AXB = \sphericalangle BXC = \sphericalangle CXA = 120°$, i budujemy drogi XA, XB i XC. Czasem więc opłaca się wyjść poza schemat prowadzenia dróg bezpośrednio między miastami i zbudować dodatkowe skrzyżowanie. To dodatkowe skrzyżowanie nazywa się często **punktem Steinera**.



Rys. 1. Punkt X jest punktem Steinera dla wierzchołków A, B, C.

Przełóżmy teraz problem księcia Piotrusia na język teorii grafów. Mamy dany spójny graf nieskierowany $G = (V, E)$, w którym każda krawędź $e \in E$ ma swoją długość $d_e \geqslant 0$. Graf to cały świat księcia Piotrusia: wierzchołki grafu odpowiadają możliwym skrzyżowaniom, a krawędzie możliwym drogom. Księstwo księcia Piotrusia to podzbiór wierzchołków $T \subseteq V$: te wierzchołki reprezentują miasta, które chcemy połączyć drogami. Mamy znaleźć najtańszą sieć dróg – czyli spójny podgraf grafu G – która łączy wszystkie miasta – czyli wierzchołki zbioru T. Zauważmy, że ten najtańszy podgraf zawsze będzie drzewem. Zbiór T często nazywa się zbiorem **terminali**. Tak określone zagadnienie to problem znalezienia najtańszego drzewa Steinera dla zbioru terminali T. Przykładowy graf z czterema terminalami znajduje się na rysunku 2.

Będziemy też czasem mówili o problemie drzewa Steinera bez długości; wtedy zakładamy, że każda krawędź grafu ma długość jeden. Zauważmy, że odpowiada to znalezieniu jak najmniejszego zbioru wierzchołków $X \subseteq V \setminus T$ tak, by zbiór $X \cup T$ *indukował* spójny podgraf grafu G, tzn. aby krawędzie, których oba końce znajdują się w tym zbiorze, wystarczały do tego, aby z każdego terminalu dało się dojść do każdego innego terminalu. Zbiór X reprezentuje dodatkowe wierzchołki poza T w poszukiwanym drzewie Steinera. Jeśli weźmiemy $|X|$ dodatkowych wierzchołków, to drzewo rozpinające $X \cup T$ będzie miało $|X| + |T| - 1$ krawędzi i taki sam będzie koszt budowy dróg, co oznacza, że minimalizując $|X|$, minimalizujemy sumę długości krawędzi wybranych do drzewa Steinera.



Rys. 2. Cztery zaznaczone wierzchołki to terminale. Pozostałe wierzchołki oraz krawędzie to możliwe skrzyżowania i drogi. Pogrubionymi kreskami zaznaczono krawędzie najtańszego drzewa Steinera, którego koszt to 14.

Problem drzewa Steinera, nawet w wersji bez długości, jest NP-trudny, co oznacza, że najprawdopodobniej nie da się efektywnie znajdować optymalnych rozwiązań tego zagadnienia. Wobec tego będziemy próbowali radzić sobie z nim w dwojaki sposób: dla większych instancji (tj. większych danych wejściowych problemu) wystarczy nam rozwiązanie przybliżone, natomiast rozwiązanie dokładne będziemy znajdować w czasie wykładniczym dla grafów na tyle dużych, na ile pozwoli nam złożoność naszego algorytmu oraz moc obliczeniowa komputera.

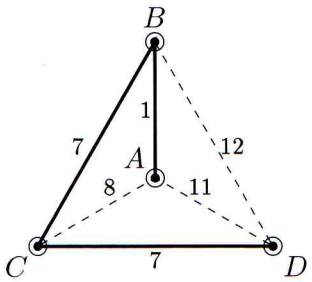
Zacznijmy od najbardziej klasycznej metody radzenia sobie z problemami NP-trudnymi, czyli od algorytmów aproksymacyjnych. Chcemy szybko – w czasie wielomianowym od rozmiaru grafu – znaleźć drzewo Steinera, które niekoniecznie będzie najlepsze, ale będzie niewiele gorsze od optymalnego.

Aby otrzymać prosty algorytm aproksymacyjny, wróćmy do oryginalnego problemu księcia Piotrusia na płaszczyźnie. Wymyślenie, by dodać punkt X, było dość trudne. A o ile gorzej byłoby, gdyby książę Piotruś zbudował drogi wzdłuż dwóch krótszych boków trójkąta ABC? Otóż okazuje się, że niewiele gorzej.

Przeanalizujmy ten pomysł w języku teorii grafów. Zmierzmy najkrótsze odległości między każdą parą miast. Zbudujmy graf pełny H o zbiorze wierzchołków T. Dla miast $s, t \in T$ jako długość krawędzi st przyjmijmy długość najkrótszej ścieżki między s i t w grafie G. Graf H odpowiada grafowi składającemu się z boków trójkąta ABC w rozpatrywanym na początku problemie na płaszczyźnie, przy założeniu, że robotnicy księcia Piotrusia nie potrafią budować dodatkowych skrzyżowań, ale potrafią budować najkrótsze drogi między każdą parą miast. W grafie H znajdźmy minimalne drzewo rozpinające D – można to zrobić wielomianowo za pomocą jednego z klasycznych algorytmów, np. Kruskala lub Prima. Następnie dla każdej krawędzi st drzewa D, w grafie G zbudujmy ciąg dróg odpowiadający

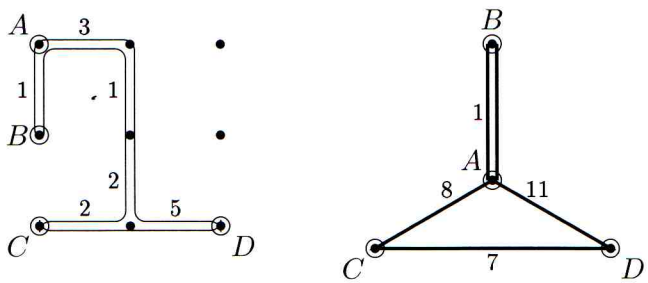
[*]Instytut Informatyki, Uniwersytet Warszawski

najkrótszej ścieżce między s i t. W ten sposób w G połączymy wszystkie miasta. Zauważmy, że możliwa jest sytuacja, w której jedną drogę (krawędź oryginalnego grafu) będziemy chcieli wybudować więcej niż raz, gdyż była ona częścią kilku najkrótszych ścieżek wybranych do drzewa rozpinającego. Jednakże w takim wypadku uzyskane drzewo Steinera będzie tańsze niż drzewo D, więc jest to dla nas sytuacja jak najbardziej korzystna. Zastanówmy się, o ile droższe może być drzewo D od rozwiązania optymalnego?



Rys. 3. Graf H otrzymany poprzez wyznaczenie najkrótszych ścieżek pomiędzy każdą parą terminali grafu z rysunku 2. Pogrubione krawędzie to minimalne drzewo rozpinające D o koszcie 15.

Niech $D^\star$ będzie optymalnym drzewem Steinera. Wykonujemy następującą operację, kluczową dla całej analizy, mianowicie podwajamy krawędzie drzewa $D^\star$. W ten sposób otrzymujemy graf spójny, w którym każdy wierzchołek ma stopień parzysty. Ma on więc cykl Eulera o długości równej dwukrotności kosztu drzewa $D^\star$. Zauważmy, że fragmenty cyklu prowadzące pomiędzy kolejnymi terminalami mają długości co najmniej takie, jak długości odpowiednich krawędzi w grafie H, gdyż długości krawędzi w grafie H odpowiadają najkrótszym ścieżkom w G. Zatem podwojony koszt drzewa $D^\star$ jest nie mniejszy niż koszt pewnego zbioru krawędzi F rozpinającego graf H, który to koszt jest z kolei nie mniejszy niż koszt drzewa D, które jest minimalnym drzewem rozpinającym w grafie H. Stąd, nasz algorytm aproksymacyjny znajdzie drzewo Steinera o koszcie nie większym niż dwukrotność kosztu optymalnego drzewa Steinera.



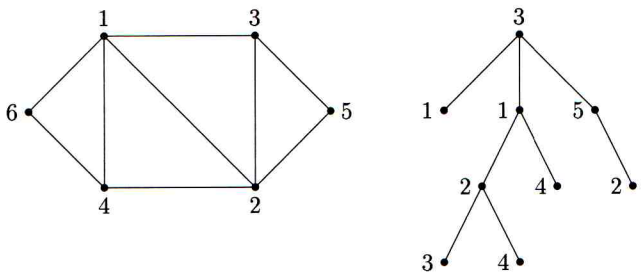
Rys. 4. Lewy rysunek przedstawia cykl Eulera otrzymany przez podwojenie krawędzi drzewa $D^\star$ z rysunku 2. Po prawej stronie znajduje się zbiór krawędzi F rozpinający graf H odpowiadający temu cyklowi Eulera (kolejne terminale na cyklu łączymy krawędziami w H).

A czy da się lepiej? To znaczy, czy w czasie wielomianowym możliwe jest skonstruowanie drzewa Steinera, którego koszt będzie mniejszy niż dwukrotność optymalnego rozwiązania?

Okazuje się, że próbując lokalnie poprawiać otrzymane drzewo, np. starając się wybierać wierzchołki Steinera o stopniu 3, można otrzymać trochę lepszy współczynnik aproksymacji (największy iloraz znalezionego przez nas rozwiązania oraz rozwiązania optymalnego). Najlepszy aktualnie znany algorytm aproksymacyjny dla problemu drzewa Steinera znajduje zawsze rozwiązanie nie gorsze niż 1,39 optymalnego kosztu. Algorytm ten pochodzi z 2010 roku, a jednym z jego autorów jest Jarosław Byrka z Uniwersytetu Wrocławskiego.

Na dziś starczy algorytmów aproksymacyjnych. Teraz będziemy próbowali rozwiązać problem drzewa Steinera w sposób dokładny, lecz w wersji bez długości. W tym sformułowaniu problem przejawia naturę kombinatoryczną.

Uporządkowanym etykietowanym drzewem nazwiemy drzewo z wyróżnionym korzeniem, w którym zbiór synów każdego wierzchołka jest uporządkowany (tzn. ich kolejność ma znaczenie). Każdy wierzchołek takiego drzewa ma etykietę będącą identyfikatorem wierzchołka z grafu G. Aby uporządkowane etykietowane drzewo D było poprawne, musi być spełniony warunek, że jeśli w drzewie D mamy krawędź łączącą wierzchołki o etykietach u, v, to w grafie G istnieje krawędź uv. Zauważmy, że nie wymagamy, aby etykiety wierzchołków drzewa były różne, jednakże etykieta każdego wierzchołka jest różna od etykiety ojca, gdyż w grafie G nie ma pętli.



Rys. 5. Graf G oraz jedno z poprawnych uporządkowanych etykietowanych drzew D. Zauważmy, że drzewo D nie musi zawierać wszystkich etykiet wierzchołków z grafu G.

Zauważmy, że jeśli w grafie G istnieje uporządkowane etykietowane drzewo D, którego zbiór etykiet zawiera identyfikatory wszystkich terminali, to optymalne drzewo Steinera ma nie więcej niż $|D|$ wierzchołków, gdyż możemy użyć drzewa D do konstrukcji zbioru dróg, które rozpinają cały zbiór terminali.

Wygodniej będzie, gdy zamienimy problem optymalizacyjny, jakim jest poszukiwanie minimalnej liczby wierzchołków w drzewie Steinera, na problem decyzyjny, w którym mamy stwierdzić, czy w danym grafie istnieje drzewo Steinera zawierające nie więcej niż d wierzchołków. Zamiast szukać drzewa Steinera, będziemy szukać uporządkowanego etykietowanego drzewa D, którego zbiór etykiet zawiera wszystkie identyfikatory terminali ze zbioru T. Okazuje się jednak, że dużo łatwiej szuka się drzewa, które nie zawiera pewnych etykiet, dlatego też użyjemy zasady włączeń i wyłączeń (czyli uogólnienia wzoru $|A \cup B| = |A| + |B| - |A \cap B|$ na większą liczbę zbiorów). Niech $N(d, X)$ dla $d \geqslant 1$ i $X \subseteq T$ oznacza liczbę uporządkowanych etykietowanych drzew o d wierzchołkach, których zbiór etykiet nie zawiera identyfikatorów terminali ze zbioru X. Wartość $N(d, X)$ dla ustalonych d oraz X można obliczyć w czasie wielomianowym standardowym algorytmem programowania dynamicznego, co pozostawiamy Drogiemu Czytelnikowi jako ćwiczenie. Zauważmy też, że na mocy zasady włączeń i wyłączeń liczba uporządkowanych drzew d-wierzchołkowych zawierających wszystkie etykiety ze zbioru T jest równa:

$$\sum_{X \subseteq T} (-1)^{|X|} \cdot N(d, X).$$

Skoro każdą wartość $N(d, X)$ potrafimy obliczyć w czasie wielomianowym, to powyższą sumę możemy obliczyć w czasie $2^{|T|} \cdot \text{poly}(|V|)$, przy czym przez $\text{poly}(|V|)$ oznaczamy czas wielomianowy względem $|V|$. Ten algorytm zaprezentował w 2009 roku Jesper Nederlof, a główną zaletą tej metody jest to, że zależność wykładnicza dotyczy jedynie $|T|$, a nie całego $|V|$. Pytaniem otwartym jest, czy da się to zrobić szybciej, np. w czasie $(2-\varepsilon)^{|T|}\text{poly}(|V|)$ dla jakiegoś $\varepsilon > 0$.

W artykule „Pokrycie wierzchołkowe kontratakuje" (*Delta* 4/2010) przedstawiliśmy ideę kernelizacji. Algorytm kernelizacyjny to taki, który szybko (w czasie wielomianowym) istotnie zmniejsza rozmiar problemu w ten sposób, że dla zmniejszonej instancji wynik jest taki sam jak dla oryginalnej. Spróbujmy zrozumieć, czego oczekiwalibyśmy od algorytmu kernelizacyjnego dla problemu drzewa Steinera. Wejściem (decyzyjnej wersji) problemu drzewa Steinera bez długości jest graf $G = (V, E)$, zbiór terminali (miast) $T \subseteq V$ oraz liczba całkowita d; chcemy znaleźć drzewo D o co najwyżej d wierzchołkach, zawierające wszystkie terminale. Chcielibyśmy w czasie wielomianowym zredukować rozmiar grafu; powiedzmy, że oczekujemy, że po redukcji graf będzie wielkości wielomianowej względem $k = |T|$. Pokażemy, dlaczego jest to niemożliwe (przy odpowiednich założeniach teoriozłożonościowych).

Na chwilę przenieśmy się do innego zagadnienia. W problemie znajdowania *motywu* w grafie mamy dany graf H, w którym każdy wierzchołek jest pomalowany na jeden z k kolorów. Chcemy wybrać po jednym wierzchołku każdego koloru tak, by wybrane wierzchołki tworzyły (indukowały) graf spójny. Taki zbiór wierzchołków nazywamy *motywem*. Co może wydać się zaskakujące, problem ten jest NP-trudny, nawet gdy H jest drzewem, którego wszystkie wierzchołki mają stopień nie większy niż trzy.

Zauważmy, że problem drzewa Steinera jest co najmniej tak trudny jak problem znajdowania motywu w grafie. Faktycznie, istnieje sprowadzenie (redukcja) przekształcające instancje problemu znajdowania motywu w grafie na instancje problemu drzewa Steinera. Mając daną instancję problemu znajdowania motywu w grafie H, dla każdego koloru i dodajemy terminal t_i połączony ze wszystkimi wierzchołkami koloru i. Łatwo zauważyć, że drzewo Steinera o $2k$ wierzchołkach w grafie H z dodanymi terminalami odpowiada motywowi w oryginalnym grafie H. Jest tak dlatego, że dwa różne terminale nigdy nie mają wspólnego sąsiada, co oznacza, że do drzewa Steinera musimy wybrać co najmniej k wierzchołków niebędących terminalami. Jednakże szukamy drzewa o co najwyżej $2k$ wierzchołkach, co oznacza, że zbiór nieterminali, które wybierzemy do drzewa Steinera, musi być spójny, gdyż nie możemy sobie pozwolić na wybranie żadnego dodatkowego wierzchołka. Ten spójny zbiór nieterminali odpowiada motywowi w oryginalnym grafie, jako że sąsiedzi różnych terminali mają różne kolory.

Teraz załóżmy, że mamy algorytm kernelizacyjny dla problemu drzewa Steinera w grafie bez długości, tj. algorytm, który w czasie wielomianowym redukuje rozmiar grafu G do wielkości wielomianowej od k – liczby terminali w grafie. Możemy wtedy wykonać następującą operację:

1. Załóżmy, że mamy dany długi ciąg instancji problemu znajdowania motywu w grafie $H_1, H_2, \ldots, H_t$; każda instancja ma dokładnie k kolorów.

2. Tworzymy graf H, który jest sumą rozłączną wszystkich grafów H_i, $1 \leqslant i \leqslant t$. Kolory wierzchołków są takie same jak w grafach H_i. Zauważmy, że w grafie H istnieje motyw wtedy i tylko wtedy, gdy istnieje on w co najmniej jednym z grafów H_i: motyw musi być spójny, więc musi zawierać się w jednej spójnej składowej grafu H.

3. Redukujemy problem znajdowania motywu w grafie H do problemu znajdowania drzewa Steinera, tak jak w poprzednim akapicie: dodajemy po jednym terminalu każdego koloru. Otrzymujemy graf G, w którym szukamy drzewa Steinera o $2k$ wierzchołkach.

4. Na grafie G uruchamiamy algorytm kernelizacyjny, który zmniejsza graf G do rozmiarów wielomianowych względem $2k$ (czyli wielomianowych względem k).

Przeanalizujmy, co się stało. Liczba początkowych instancji – t – mogła być bardzo duża, ponadwielomianowa w stosunku do k. Na końcu otrzymaliśmy jedną instancję problemu znajdowania drzewa Steinera o dużo mniejszej liczbie wierzchołków – wielomianowej względem k. Przy tym ta końcowa instancja jest „syntezą" początkowych instancji: istnieje w niej odpowiednio małe drzewo Steinera wtedy i tylko wtedy, gdy co najmniej jedna początkowa instancja miała motyw. Końcowa instancja jest jednak bardzo mała i nie ma szans pomieścić informacji o wszystkich początkowych instancjach problemu znajdowania motywu w grafie – intuicyjnie oznacza to, że musieliśmy większość instancji H_i w jakiś sposób rozwiązać. Okazuje się, że dla problemu NP-zupełnego taka operacja nie może nam się udać; powyższe rozumowanie można sformalizować i pokazać, że algorytm kernelizacyjny dla problemu drzewa Steinera pociągałby za sobą dużą rewolucję w teorii złożoności, co jest mało prawdopodobne. Dodajmy tylko, że użyta tutaj technika jest również stosunkowo nowa – pierwsi użyli jej Lance Fortnow i Rahul Santhanam w swojej pracy z 2008 roku.

Jak widać, problem drzewa Steinera ma wiele ciekawych obliczy i każdy znajdzie w nim coś dla siebie. Barwne jest życie algorytmika.

Do grupy naukowej badającej różne aspekty problemów NP-trudnych na wydziale Matematyki, Informatyki i Mechaniki Uniwersytetu Warszawskiego należą dr Łukasz Kowalik, dr Marcin Mucha, dr hab. Piotr Sankowski, dr Jakub Wojtaszczyk, a także doktoranci i studenci wydziału, w tym autorzy tego artykułu. Główne kierunki badań stanowią aproksymacja oraz algorytmy dokładne dla problemów NP-trudnych. Prace, których autorami są członkowie zespołu, są publikowane na najlepszych międzynarodowych konferencjach poświęconych informatyce teoretycznej oraz w uznanych periodykach naukowych. Członkowie grupy są laureatami programów stypendialnych Fundacji Nauki Polskiej, miesięcznika *Polityka* i nagród im. Witolda Lipskiego. Ponadto prace naukowe grupy badawczej prowadzone są w ramach grantów MNiSW, jak również prestiżowego grantu „Starting Independent Researcher Grant" ufundowanego przez European Research Council, którego kierownikiem jest dr hab. Piotr Sankowski.

Objaśnienie terminologii grafowej wykorzystanej w tym artykule można znaleźć np. w książce R.J. Wilsona *Wprowadzenie do teorii grafów*, natomiast opis wspomnianych algorytmów Kruskala i Prima służących do wyznaczania minimalnego drzewa rozpinającego – w książce T.H. Cormen, C.E. Leiserson, R.L. Rivest, C. Stein, *Wprowadzenie do algorytmów*.