Uniwersytet Jagielloński w Krakowie

Wydział Matematyki i Informatyki

Pola Kyzioł

Nr albumu: 1092406

Algorytmy dynamiczne po dekompozycji drzewowej dla problemów grafowych o spójnych rozwiązaniach.

Praca magisterska
na kierunku Informatyka Analityczna

Praca wykonana pod kierunkiem
dr hab. Tomasz Krawczyk
Instytut Informatyki Analitycznej

Kraków 2019

Oświadczenie autora pracy

Świadom odpowiedzialności prawnej oświadczam, że niniejsza praca dyplomowa została napisana przeze mnie samodzielnie i nie zawiera treści uzyskanych w sposób niezgodny z obowiązującymi przepisami.

Oświadczam również, że przedstawiona praca nie była wcześniej przedmiotem procedur związanych z uzyskaniem tytułu zawodowego w wyższej uczelni.

.....	.....
Kraków, dnia	Podpis autora pracy

Oświadczenie kierującego pracą

Potwierdzam, że niniejsza praca została przygotowana pod moim kierunkiem i kwalifikuje się do przedstawienia jej w postępowaniu o nadanie tytułu zawodowego.

.....	.....
Kraków, dnia	Podpis kierującego pracą

Spis treści

Rozdział 1

Wprowadzenie

Problem Cyklu Hamiltona oraz *problem Drzewa Steinera* są jednymi z najstarszych i najczęściej badanych problemów należących do klasy problemów $\mathcal{NP}$-trudnych. W niniejszej pracy rozważam warianty decyzyjne tych problemów, których definicje przedstawiam poniżej.

Problem Cyklu Hamiltona
Wejście: Graf G.
Wyjście: Czy istnieje cykl prosty o długości $|V(G)|$.

Problem Drzewa Steinera
Wejście: Graf $G(V, E)$, zbiór wierzchołków terminalnych $K \subset V(G)$, liczba naturalna ℓ.
Wyjście: Czy istnieje spójny podgraf $H \subset G$ taki, że $K \subset V(H)$ oraz $|E(H) \leq \ell|$.

Dla powyższych problemów istnieją algorytmy dynamiczne, parametryzowane po szerokości tzw. *dekompozycji drzewowej*, które należą do klasy problemów $\mathcal{FPT}$. Złożoność czasowa ich klasycznych wersji zależy od czynnika $k^{O(k)}$, gdzie k jest *szerokością drzewową grafu* G. Definicje dekompozycji drzewowej, jak i szerokości drzewowej są przedstawione w rozdziale 2. W *Parametrized Algorithms* [1] autorzy opisują algorytmy probabilistyczne, które zbijają tę zależność do $2^{O(k)}$. Algorytmy te bazują na technice *Cut & Count*, która redukuje problem decyzyjny do problemu zliczania rozwiązań modulo 2. Dokładnie jest ona opisana w rozdziale 4.

Celem niniejszej pracy jest opis, analiza, implementacja oraz porównanie wydajności klasycznych algorytmów dynamicznych po dekompozycji drzewowej oraz algorytmów probabilistycznych wykorzystujących technikę Cut & Count dla problemów: Cyklu Hamiltona oraz Drzewa Steinera.

Wszystkie powyższe algorytmy zostały zaimplementowane w języku programowania C++ oraz przetestowane. Testy wydajnościowe zostały przeprowadzone na wygenerowanych instancjach dekompozycji drzewowych, ich wyniki są zobrazowane na wykresach przedstawionych w rozdziale 5. Implementacja algorytmów jest udostępniona w repozytorium na githubie [4]. Wszystkie instancje wejściowe dekompozycji drzewowych można wyeksportować do plików .dot, a następnie wyrenderować z nich wizualizacje drzew. Więcej informacji znajduje się w pliku README.txt na githubie.

Rozdział 2

Podstawowe definicje

Definicja 1. *Dekompozycją drzewową grafu* G nazywamy parę $\mathcal{T} = (T, \{X_t : t \in V(T)\})$, gdzie T jest drzewem, a $\{X_t : t \in V(T)\}$ rodziną zbiorów wierzchołków grafu G spełniającą następujące warunki:

(i) Dla każdej krawędzi $\{u, v\} \in E(G)$ istnieje węzeł $t \in V(T)$, taki że $u \in X_t$ i $v \in X_t$.

(ii) Dla każdego wierzchołka $v \in V(G)$ zbiór $\{t \in V(T) : v \in X_t\}$ jest poddrzewem drzewa T.

Od tej pory wierzchołki grafu G będą nazywane po prostu *wierzchołkami*, natomiast wierzchołki drzewa T będą nazywane *węzłami*.

Definicja 2. *Szerokość drzewowa dekompozycji drzewowej* $\mathcal{T}$, $sd_{\mathcal{T}}$, opisuje rozmiar najliczniejszego węzła pomniejszony o 1:

$$sd_{\mathcal{T}} = max_{t \in V(T)} |X_t - 1|$$

Definicja 3. *Szerokość drzewowa grafu* G, standardowo oznaczana przez sd_G lub k, jest minimalną szerokością drzewową wziętą po wszystkich możliwych dekompozycjach drzewowych G:

$$sd_G = min\{sd_{\mathcal{T}} : \mathcal{T} \text{ jest dekompozycją drzewową } G\}$$

Przy konstruowaniu algorytmów dynamicznych działających po dekompozycji drzewowej grafu łatwiej jest posługiwać się tzw. *ładną dekompozycją drzewową*.

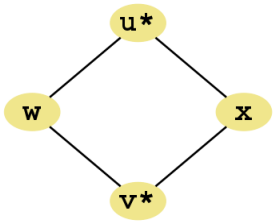
Definicja 4. *Ładna dekompozycja drzewowa* $\mathcal{T} = (T, \{X_t\}_{t \in V(T)})$ to dekompozycja drzewowa o następujących własnościach:

(i) T jest drzewem ukorzenionym.

(ii) Każdy węzeł $t \in T$ ma jeden z następujących pięciu typów:

1) t jest `LIŚCIEM`: $X_t = \emptyset$.

2) t jest węzłem `WPROWADZAJĄCYM` v: t ma o jeden wierzchołek więcej niż jego jedyne dziecko: $X_t = X_{t'} \cup \{v\}$. Każdy wierzchołek $v \in V(G)$, ma co najmniej jeden węzeł wprowadzający.



Rysunek 2.1: Przykładowy graf G.

3) t jest węzłem `ZAPOMINAJĄCYM` v: t ma o jeden wierzchołek mniej niż jego jedyne dziecko: $X_t = X_{t'} \setminus \{v\}$. Jego szczególnym reprezentantem jest korzeń. Dla każdego wierzchołka $v \in V(G)$, istnieje dokładnie jeden węzeł zapominający.

4) t jest węzłem `SCALAJĄCYM`: t posiada dwoje dzieci: $X_t = X_{t'} = X_{t''}$, scala dwa podgrafy o przecięciu X_t.

5) t jest węzłem `WPROWADZAJĄCYM KRAWĘDŹ` uv: $X_t = X_{t'}$, ten typ węzła nie pojawił się w pierwotnej definicji ładnej dekompozycji drzewowej, ale znacząco ułatwia definiowanie algorytmów, które na nich operują. Węzeł wprowadzający uzupełnia reprezentację grafu G w drzewie T o krawędź $uv \in E(G)$, $u \in X_t$ i $v \in X_t$. Dla każdego uv istnieje dokładnie jeden węzeł wprowadzający i - przyjmując bez straty ogólności, $t(u)$ jest przodkiem $t(v)$ (gdzie $t(v)$ to najwyższy węzeł, taki że $v \in X_{t(v)}$) - znajduje się on pomiędzy $t(v)$ a węzłem zapominającym v.

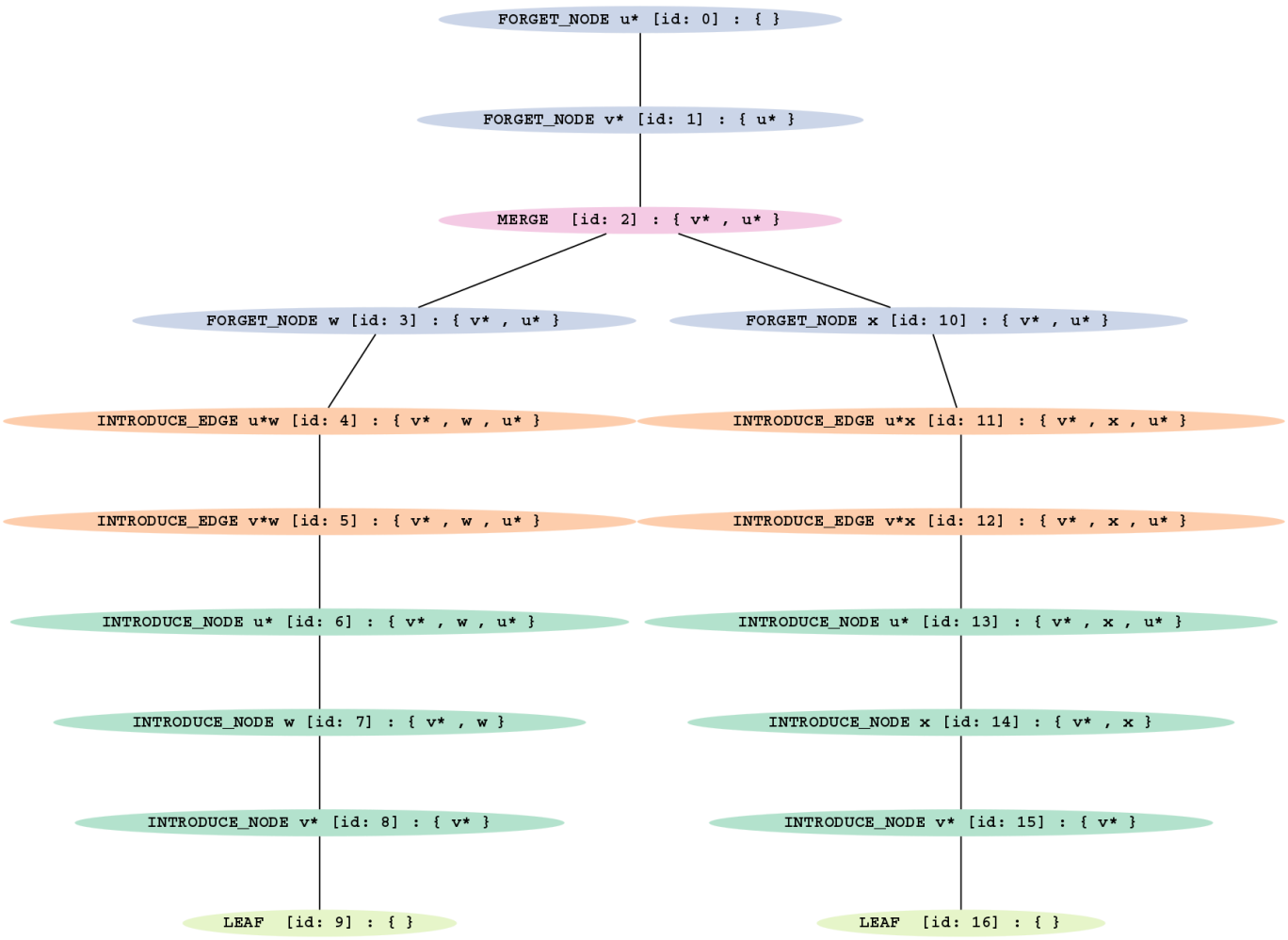
Definicja 5. Dla węzła t, definiujemy $G_t = (V_t, E_t)$, gdzie:

V_t jest zbiorem wierzchołków wprowadzonych w poddrzewie ukorzenionym w t.
E_t jest zbiorem krawędzi wprowadzonych w poddrzewie ukorzenionym w t.

Problem obliczania dekompozycji drzewowej należy do klasy problemów FPT. Bodlaender [2] jako pierwszy podał liniowy algorytm znajdowania dekompozycji drzewowej o szerokości k (o ile taka dekompozycja istnieje). Natomiast Kloks [3] pokazał, że dla każdego grafu o szerokości drzewowej k istnieje ładna dekompozycja drzewowa o szerokości drzewowej k, którą można skonstruować w czasie liniowym od ilości wierzchołków grafu G.

Definicja 6. *Problem Drzewa Steinera.* Na wejściu mamy dany graf G wraz z jego ładną dekompozycją drzewową $\mathcal{T} = (T, \{X_t\}_{t \in V(T)})$, zbiór tzw. terminali $K \subseteq V(G)$ oraz liczbę naturalną ℓ. Pytamy, czy istnieje spójny graf acykliczny, który łączy wszystkie terminale i ma co najwyżej ℓ krawędzi, potocznie zwany drzewem Steinera.

Definicja 7. *Problem Cyklu Hamiltona.* Na wejściu mamy dany graf G wraz z jego ładną dekompozycją drzewową $\mathcal{T} = (T, \{X_t\}_{t \in V(T)})$. Pytamy, czy istnieje cykl prosty, który zawiera wszystkie wierzchołki $V(G)$.



Rysunek 2.2: Ładna dekompozycja drzewowa grafu G z rys. 2.1. Wierzchołki terminalne są oznakowane *.

Rozdział 3

Klasyczne algorytmy dynamiczne

3.1 Drzewo Steinera

W tym rozdziale zostanie przedstawiony klasyczny algorytm dynamiczny po dekompozycji drzewowej dla problemu Drzewa Steinera.

W celu uproszczenia algorytmu, przyjmujemy, że każdy węzeł drzewa T zawiera przynajmniej jeden terminal. Uzyskujemy to, wybierając dowolny wierzchołek $v_0 \in K$ i dodając go do każdego węzła oraz usuwając `WPROWADZAJĄCY` v_0 i `ZAPOMINAJĄCY` v_0. Własności ładnej dekompozycji drzewowej są zachowane z modyfikacją, że liście i korzeń nie są puste.

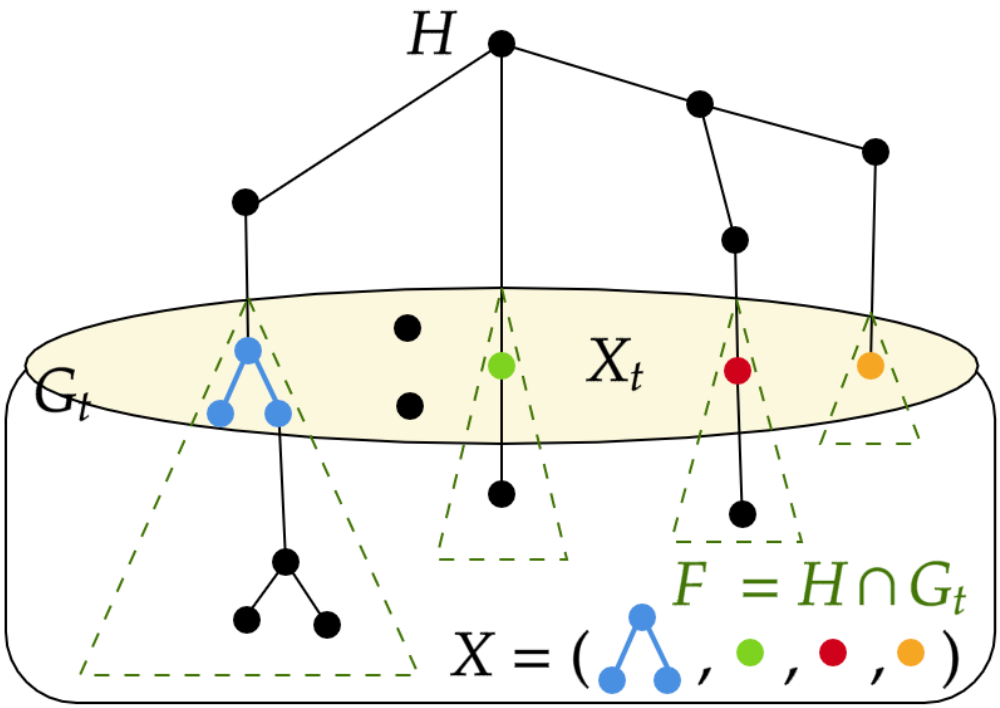
Zastanówmy się, jak wygląda ślad drzewa Steinera w węźle t, przykładowy został zobrazowany na rys. 3.1. Dla H będącego dowolnym drzewem Steinera łączącym wszystkie terminale K, przecięcie H i G_t tworzy las F, który zawiera przynajmniej jeden wierzchołek terminalny v_0. Szukane H jest spójne oraz X_t zawiera terminal, co implikuje, że każde drzewo należące do F przecina X_t. Ponadto każdy terminal $\in K \cap V_t$ należy do F. Zauważmy, że ślad drzewa Steinera $(X, \mathcal{P})$ w węźle t jest jednoznacznie zdefiniowany przez zbiór wierzchołków $X = V(F) \cap X_t$ oraz jego podział $\mathcal{P} = \{X_1, X_2, \ldots, X_z\}$ na zbiory wierzchołków przynależnych do poszczególnych drzew F.

Dla każdej trójki $(t, X, \mathcal{P})$ trzymamy w $c(t, X, \mathcal{P})$ rozmiar (liczbę krawędzi) najmniejszego F w G_t. Jeśli dla danej trójki nie istnieje poprawny ślad: $c(t, X, \mathcal{P}) = \infty$. Musimy na bieżąco śledzić partycję $\mathcal{P}$, ponieważ może się ona zmieniać wraz z każdym nowo przetwarzanym węzłem. Węzły `SCALAJĄCY` lub `UZUPEŁNIAJĄCY` uv mogą zepsuć wcześniej poprawną partycję, tworząc cykl. Wynik końcowy odpowiada wartości $c(r, \{v_0*\}, \{\{v_0*\}\})$, gdzie r jest korzeniem. Poniżej prezentuję formuły rekurencyjne obliczania wartości c ze względu na typ węzła t. Dla parametrów niezdefiniowanych poniżej, c przyjmuje wartość ∞.

t jest `LIŚCIEM`

$$c(t, \{v_0*\}, \{\{v_0*\}\}) = 0$$

t jest węzłem `WPROWADZAJĄCYM` v - zauważmy, że v nie należał do żadnego z dzieci, wobec czego krawędzie incydentne z v nie zostały jeszcze wprowadzone poprzez węzły wprowadzające i v jest wierzchołkiem izolowanym. Wierzchołek izolowany ma swój własny, jednoelementowy



Rysunek 3.1: Przykładowy ślad drzewa Steinera w węźle t.

komponent w $\mathcal{P}$. Jeśli v jest terminalem, musi należeć do X.

$$c(t, X, \mathcal{P}) = \begin{cases} c(t', X \setminus \{v\}, \mathcal{P} \setminus \{\{v\}\}) & \text{jeśli } v \in X, \\ c(t', X, \mathcal{P}) & \text{jeśli } v \notin X \wedge v \notin K. \end{cases}$$

t jest węzłem `WPROWADZAJĄCYM KRAWĘDŹ` uv - mamy 3 przypadki, które rozpatrujemy osobno:

1. $u \notin X \vee v \notin X$
2. $u \in X \wedge v \in X$ oraz u i v są w różnych komponentach $\mathcal{P}$ ($u \in X_i$, $v \in X_j$, $i \neq j$)
3. $u \in X \wedge v \in X$ oraz u i v są w tych samych komponentach $\mathcal{P}$ ($u, v \in X_i$)

W przypadkach 1 oraz 2 krawędź uv nie może należeć do rozwiązania. Natomiast w przypadku 3 mamy dwie możliwości - albo włączamy krawędź do rozwiązania, albo nie. Jeśli nie dodajemy krawędzi do rozwiązania, przepisujemy wynik dla podproblemu $G_{t'}$ dla tej samej partycji. W przeciwnym przypadku, nowo dodawana krawędź musiała połączyć dwa rozłączne bloki partycji t'. Ponieważ optymalizujemy po rozmiarze śladu, iterujemy się po wszystkich partycjach $\mathcal{P}'$ węzła t', gdzie u ($\in X_i$) i v ($\in X_j$) nie należą do tego samego komponentu ($i \neq j$) (inaczej powstałby cykl), ale po połączeniu X_i z X_j dają partycję $\mathcal{P}$. Możemy to zapisać następująco:

$$c(t, X, \mathcal{P}) = \min \left\{ \min_{\mathcal{P}'} c(t', X, \mathcal{P}') + 1, \quad c(t', X, \mathcal{P}) \right\}$$

t jest węzłem `ZAPOMINAJĄCYM` v - wierzchołek v mógł należeć do śladu w węźle t'. Dlatego musimy popatrzeć na te partycje $\mathcal{P}'$ węzła t', które go zawierają, a po jego usunięciu dają nam $\mathcal{P}$. Jednakże tylko na te, w których nie jest on singletonem, w przeciwnym przypadku nasz ślad nigdy nie stałby się spójnym drzewem Steinera (wszystkie krawędzie incydentne z zapominanym wierzchołkiem są wprowadzane przed jego zapomnieniem i tylko wtedy mogą one zostać dodane do rozwiązania). Jednocześnie musimy pamiętać, że wierzchołek v, który nie jest terminalem, może nie należeć do końcowego rozwiązania - wtedy przepisujemy wynik dla t', nie zmieniając parametrów. Kluczowa dla tego przypadku jest obserwacja, że jeśli v jest terminalem, nie istnieją rozwiązania dla podproblemu $G_{t'}$ nie uwzględniające v (tzn. ich wynikiem jest ∞).

$$c(t, X, \mathcal{P}) = \min \left\{ \min_{\mathcal{P}'} c(t', X \cup \{v\}, \mathcal{P}'), \quad c(t', X, \mathcal{P}) \right\}$$

t jest węzłem `SCALAJĄCYM` - dla węzła scalającego łączymy rozwiązania podproblemów $G_{t'}$ i $G_{t''}$. Poniżej znajduje się kilka spostrzeżeń kluczowych przy definiowaniu formuły rekurencyjnej dla węzła scalającego:

(a) $V_{t'} \cap V_{t''} = X$

(b) $E_{t'} \cap E_{t''} = \emptyset$, ponieważ krawędzie wprowadzane są najpóźniej jak to możliwe, tj. przed węzłami zapominającymi. Zakładając nie wprost, że istnieje krawędź uv zarówno w $E_{t'}$ jak i w $E_{t''}$, muszą również istnieć dwa węzły zapominjące u lub v. Prowadzi to do sprzeczności, ponieważ dla każdego wierzchołka może istnieć tylko jeden węzeł zapominający (z definicji dekompozycji drzewowej).

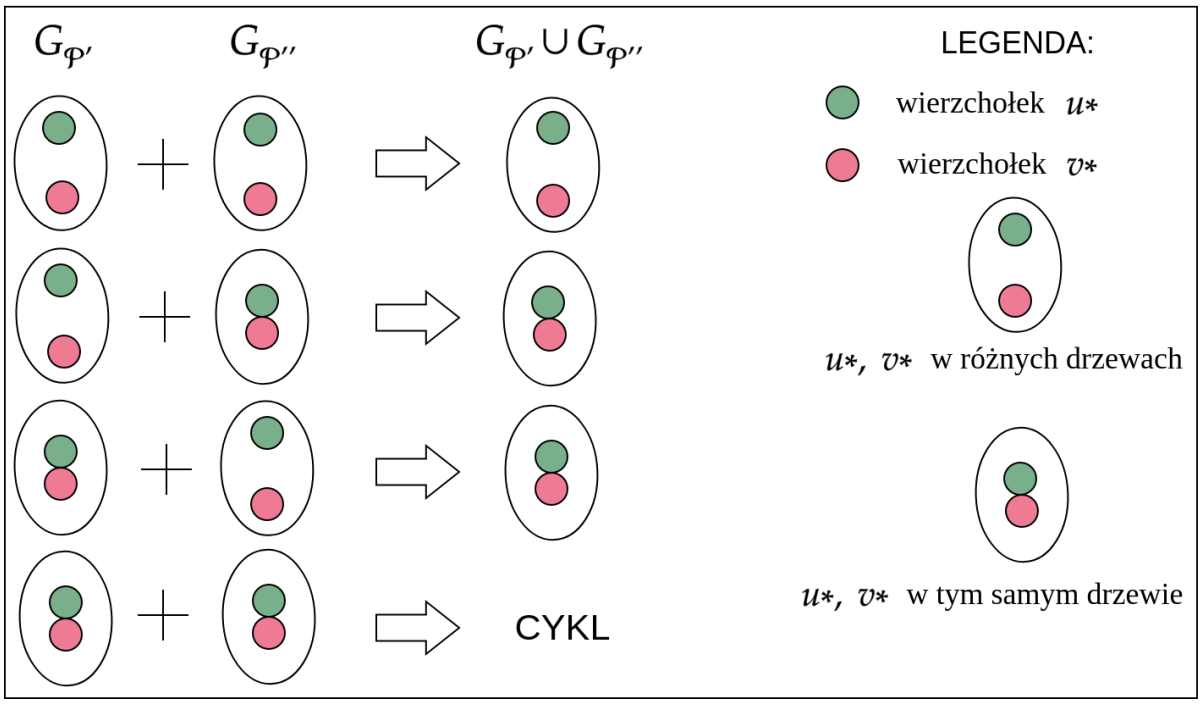
(c) Połączenie $G_{t'}$ z $G_{t''}$ może zawierać cykl. By uniknąć cykli autorzy algorytmu wprowadzają pomocniczą strukturę $G_{\mathcal{P}}$, która jest lasem o zbiorze wierzchołków X oraz której drzewa korespondują z podziałem $\mathcal{P}$. Problem łączenia dwóch śladów (dla t' oraz dla t'') sprowadza się do problemu łączenia dwóch lasów $G_{\mathcal{P}'} \cup G_{\mathcal{P}''}$. Zauważmy, że z perspektywy wyliczania poprawnego rozwiązania dla węzła t, nie jest ważne, jaki kształt mają poszczególne drzewa, istotny jest fakt, że drzewo jest grafem spójnym (między dowolną parą wierzchołków do niego należących, istnieje ścieżka).

Warunki, które muszą zostać spełnione przy scalaniu dwóch rozwiązań są następujące:

(i) $G_{\mathcal{P}'} \cup G_{\mathcal{P}''}$ nie zawiera cyklu.

(ii) $G_{\mathcal{P}'} \cup G_{\mathcal{P}''}$ odpowiada $G_{\mathcal{P}}$ z dokładnością do kształtu poszczególnych drzew.

W implementacji powyższego algorytmu do reprezentowania problemu łączenia lasów $G_{\mathcal{P}'}$, $G_{\mathcal{P}''}$ wykorzystałam strukturę zbiorów rozłącznych z łączeniem według rangi i kompresją ścieżek. Dzięki niej łatwo wykryć cykl oraz zbadać, które wierzchołki są w tych samych, spójnych komponentach $\mathcal{P}$. Rysunek 3.2 przedstawia wszystkie możliwe lasy $G_{\mathcal{P}'}$, $G_{\mathcal{P}''}$ dla



Rysunek 3.2: Rezultaty otrzymane w wyniku połączenia lasów $G_{\mathcal{P}'}$, $G_{\mathcal{P}''}$ w węźle scalającym z rys. 2.2.

węzła scalającego z rysunku 2.2 wraz z rezultatami połączenia lasów. Końcowe rozwiązanie wyliczamy na podstawie poniższego wzoru:

$$c[t][X][\mathcal{P}] = \min_{\mathcal{P}',\mathcal{P}''} c[t'][X][\mathcal{P}'] + c[t''][X][\mathcal{P}'']$$

Przedstawione zostały formuły rekurencyjne dla wszystkich typów węzłów. Przejdźmy zatem do wyliczenia złożoności czasowej standardowego algorytmu dynamicznego po dekompozycji drzewowej dla problemu drzewa Steinera. Poniżej znajdują się istotne spostrzeżenia:

(a) Każdy węzeł ma co najwyżek $k+2$ wierzchołki.

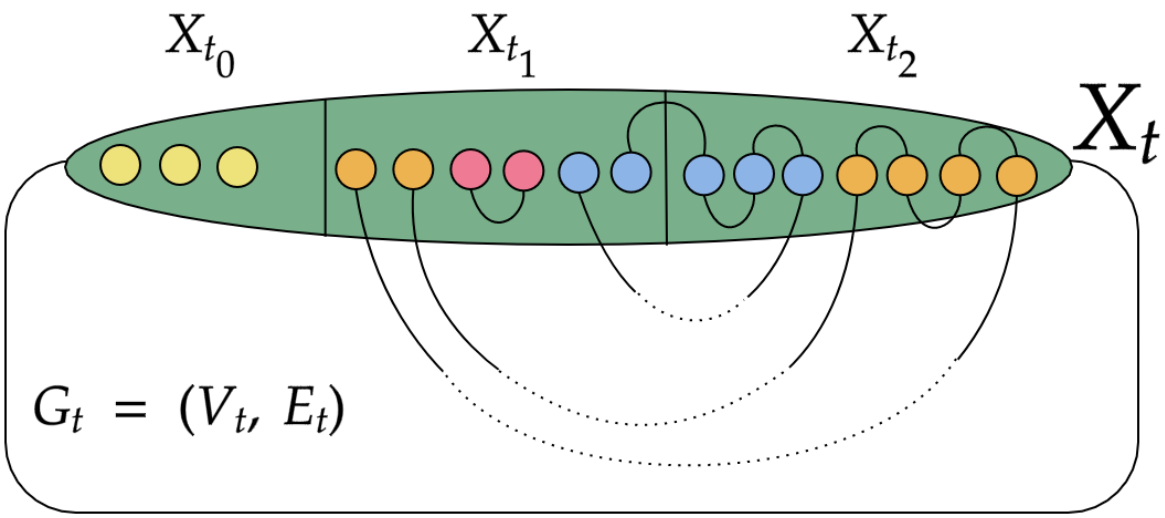
(b) Wszystkich $X \subseteq X_t$ jest $2^{|X_t|}$ a zatem nie więcej niż 2^{k+2}.

(c) Wszystkich partycji X jest $|X|^{|X|}$ a zatem nie więcej niż $(k+2)^{k+2}$

(d) Dla każdego węzła mamy $2^{k+2} \cdot (k+2)^{k+2} = k^{O(k)}$ stanów.

(e) Wartości funkcji c dla każdego węzła wyliczamy w czasie $(k^{O(k)})^2$.

Twierdzenie 1. *Mając dany n-wierzchołkowy graf G razem ze zbiorem terminali $K \subseteq V(G)$ oraz dekompozycją drzewową o szerokości drzewowej nie większej niż k, można wyliczyć rozmiar minimalnego drzewa Steinera w czasie $k^{O(k)} \cdot n$.*



Rysunek 3.3: Przykładowy ślad ścieżki Hamiltona dla węzła t.

3.2 Cykl Hamiltona

W tym rozdziale przedstawię klasyczny algorytm dynamiczny po dekompozycji drzewowej dla problemu istnienia cyklu Hamiltona.

Zakładamy, że mamy daną ładną dekompozycję drzewową lekko zmodyfikowanego grafu $G' : \mathcal{T} = (T, \{X_t\}_{t \in V(T)})$. Modyfikacja polega na tym, że kopiujemy wierzchołek v_r, będący w korzeniu dekompozycji drzewowej grafu G (`ZAPOMINAJĄCY` v_r) wraz z jego krawędziami, tj. dla każdej krawędzi $uv_r \in E(G)$, dostajemy dwie krawędzie uv_{r_1} i uv_{r_2}, gdzie v_{r_1} jest byłym wierzchołkiem v_r, a v_{r_2} jego kopią. Z dekompozycji drzewowej $\mathcal{T}$ grafu G' usuwamy węzły: zapominających v_{r_1} oraz zapominający v_{r_2}. Problem szukania cyklu Hamiltona modyfikujemy do problemu szukania ścieżki Hamiltona o końcach v_{r_1} i v_{r_2}.

Ślad cyklu Hamiltona w poddrzewie węzła t składa się ze zbioru rozłącznych ścieżek $F = H \cap G_t = \{C_1, C_2, \ldots, C_z\}$, gdzie H jest szukanym cyklem. F poza liśćmi nie jest puste, ponieważ wszystkie wprowadzone wierzchołki muszą należeć do śladu. Szukane H jest spójne, co implikuje, że każda ścieżka $C_1, C_2, \ldots, C_z$ należąca do F przecina X_t. Ponadto musi ona przecinać X_t oboma swoimi końcami, w przeciwnym przypadku nie dostalibyśmy spójnego rozwiązania końcowego. Z powyższych warunków wynika, że wierzchołki X_t możemy sklasyfikować następująco (rys. 3.3):

- X_{t_0} zbiór wierzchołków izolowanych w F (o stopniu 0)
- X_{t_1} zbiór wierzchołków będących końcami ścieżek $C_1, C_2, \ldots, C_z$ (o stopniu 1)
- X_{t_2} zbiór wierzchołków należących do ścieżek $C_1, C_2, \ldots, C_z$ i nie będących ich końcami (o stopniu 2)

Ślad cyklu Hamiltona w węźle t jest jednoznacznie zdefiniowany podziałem zbioru X_t na trzy podzbiory $\mathcal{D} = (X_{t_0}, X_{t_1}, X_{t_2})$ oraz dopasowaniem M. Ślad jest poprawny, jeśli spełnione są poniższe warunki:

(i) $V(F) = V_t$

(ii) $u \in X_{t_i} \Leftrightarrow u \in X_t$ i wierzchołek v ma stopień i w F

(iii) u jest skojarzone z v ($\{u, v\} \in M$) $\Leftrightarrow u \in X_{t_1} \wedge v \in X_{t_1} \wedge \exists y : u \in C_y \wedge v \in C_y$ (u i v są końcami ścieżki C_y)

Dla każdej trójki $(t, \mathcal{D}, M)$ algorytm liczy, czy wyznacza ona poprawny ślad - *true*/*false*. Musimy na bieżąco śledzić dopasowanie M, ponieważ węzły typu `SCALAJĄCY` oraz `WPROWADZAJĄCY KRAWĘDŹ` uv mogą utworzyć cykl. Wynik końcowy odpowiada wartości $c(r, (\emptyset, \{v_{r_1}, v_{r_2}\}, \emptyset), \{\{v_{r_1}, v_{r_2}\}\})$, gdzie r jest korzeniem T. Poniżej prezentuję formuły rekurencyjne obliczania wartości c ze względu na typ węzła t.

t jest `LIŚCIEM`

$$c(t, (\emptyset, \emptyset, \emptyset), \emptyset) = true$$

t jest węzłem `WPROWADZAJĄCYM` v - zauważmy, że w momencie dodawania wierzchołka jego stopień wynosi 0, ponieważ żadna z incydentnych do niego krawędzi nie została jeszcze wprowadzona:

$$c(t, (X_{t_0}, X_{t_1}, X_{t_2}), M) = \begin{cases} c(t', (X_{t_0} \setminus \{v\}, X_{t_1}, X_{t_2}), M) & \text{jeśli } v \in X_{t_0}, \\ false & \text{wpp.} \end{cases}$$

t jest węzłem `ZAPOMINAJĄCYM` v - wszystkie krawędzie incydentne do zapominanego wierzchołka zostały już wprowadzone. Ślad jest poprawny tylko wtedy, kiedy zapominany wierzchołek jest stopnia 2:

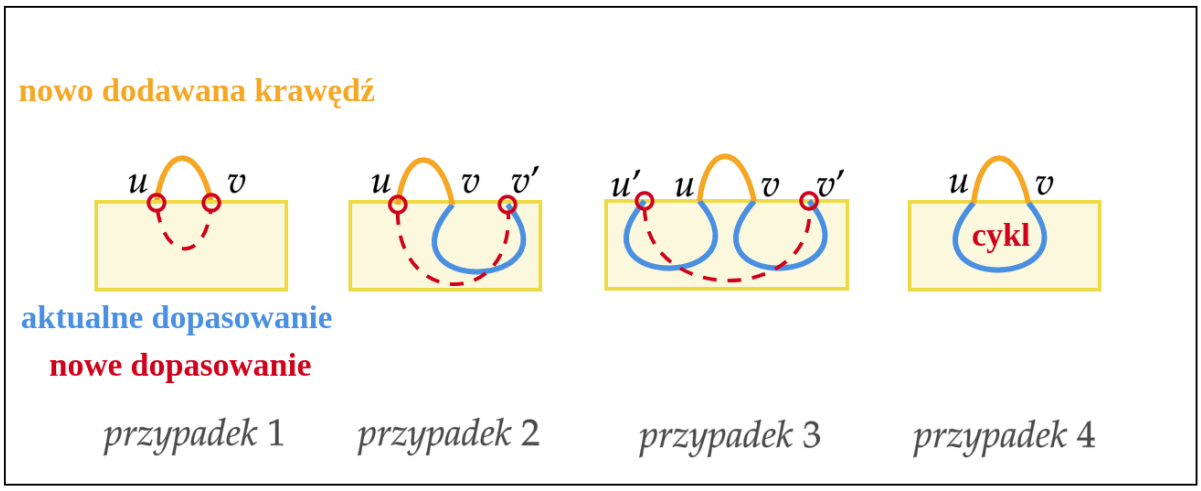
$$c(t, (X_{t_0}, X_{t_1}, X_{t_2}), M) = c(t', (X_{t_0}, X_{t_1}, X_{t_2} \cup \{v\}), M)$$

t jest węzłem `WPROWADZAJĄCYM` uv - dodawanie krawędzi incydentnej do wierzchołka zwiększa jego stopień o 1, wobec czego możemy dodać krawędź uv, jeśli $u \in X_{t_i}$: $i < 2$ oraz $v \in X_{t_j}$: $j < 2$. Nie jest to jednak warunek wystarczający, ponieważ nie gwarantuje nam, że nie dostaniemy cyklu. Jeśli $u \in X_{t_1} \wedge v \in X_{t_1}$ oraz $\exists m : m \in M \wedge m = \{u, v\}$, nie możemy wziąć krawędzi uv, ponieważ powstałby cykl. W takim przypadku wynik przepisujemy z węzła t', nie zmieniając dopasowania M. Jeśli natomiast dodanie krawędzi do rozwiązania jest możliwe, mamy alternatywę - albo jej nie dodajemy i przepisujemy wynik jak poprzednio, albo ją dodajemy. Poniżej przedstawiam formuły rekurencyjne dla wymienionych przypadków:

1. $\{u, v\} \notin M$
$$c(t, (X_{t_0}, X_{t_1}, X_{t_2}), M) = c(t', (X_{t_0}, X_{t_1}, X_{t_2}), M)$$

2. $\{u, v\} \in M$ oraz $u \in X_{t_1} \wedge v \in X_{t_1}$ (*przypadek* 1 na rys. 3.4)
$$c(t, (X_{t_0}, X_{t_1}, X_{t_2}), M) = c(t', (X_{t_0} \cup \{u, v\}, X_{t_1} \setminus \{u, v\}, X_{t_2}), M \setminus \{u, v\})$$

3. $\{u, v\} \in M$ oraz $u \in X_{t_1} \wedge v \in X_{t_2}$ (*przypadek* 2 na rys. 3.4)
$$c(t, (X_{t_0}, X_{t_1}, X_{t_2}), M) = \bigvee_{M'} c(t', (X_{t_0} \cup \{u\}, X_{t_1} \cup \{v\} \setminus \{u\}, X_{t_2} \setminus \{v\}), M \setminus \{u, v'\} \cup \{v, v'\})$$



Rysunek 3.4: Dodawanie nowej krawędzi uv do dopasowania M.

4. $\{u, v\} \in M$ oraz $u \in X_{t_2} \wedge v \in X_{t_1}$ (analogicznie jak w punkcie 3)

5. $\{u, v\} \in M$ oraz $u \in X_{t_2} \wedge v \in X_{t_2}$ (*przypadek* 3 na rys. 3.4)

$$c(t, (X_{t_0}, X_{t_1}, X_{t_2}), M) = \bigvee_{M'} c(t', (X_{t_0}, X_{t_1} \cup \{u, v\}, X_{t_2} \setminus \{u, v\}), M \setminus \{u', v'\} \cup \{u, u'\} \cup \{v, v'\})$$

6. $\{u, v\} \in M$ oraz $u \in X_{t_2} \wedge v \in X_{t_2}$ (*przypadek* 4 na rys. 3.4)

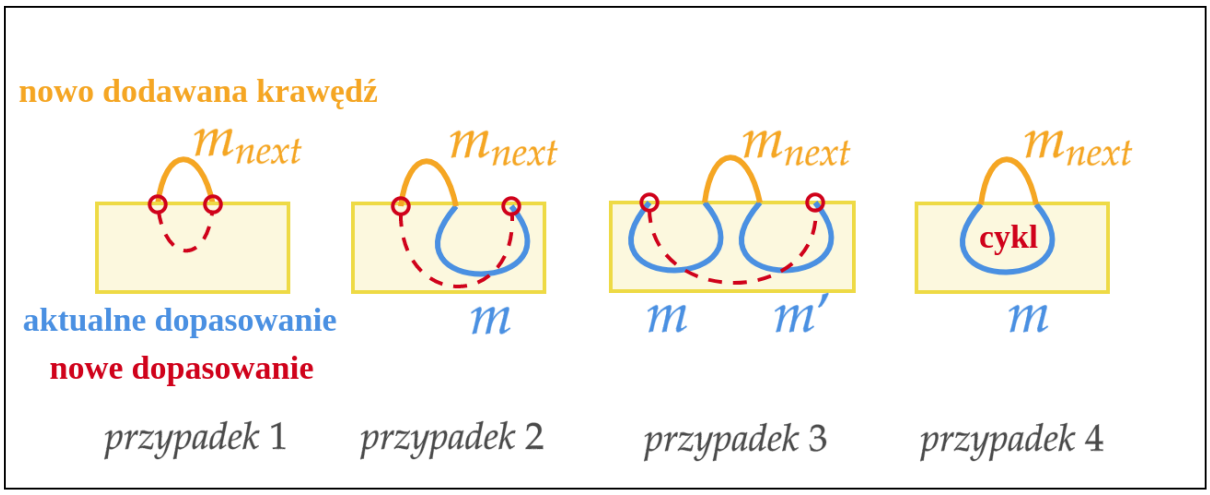
$$c(t, (X_{t_0}, X_{t_1}, X_{t_2}), M) = false$$

t jest węzłem `SCALAJĄCYM` - zauważmy, że scalane $F_{t'}$ oraz $F_{t''}$ nie mają wspólnych krawędzi, jedynie wierzchołki. Z powyższego wynika, że dla każdego wierzchołka v należącego do X_t: $deg_t(v) = deg_{t'}(v) + deg_{t''}(v)$. Zauważmy, że musi być spełniony warunek $deg_t(v) \leq 2$. Jeśli $\mathcal{D}_{t'}$ i $\mathcal{D}_{t''}$ nie spełniają tego warunku, nie rozpatrujemy ich. Jednakże nie jest to warunek wystarczający, by wartość dla węzła t była poprawna. Może się zdarzyć (jak przy dodawaniu krawędzi), że scalanie utworzy nam cykl. Obliczając nowe dopasowanie, dodajemy dopasowania $m_{next} = \{m_{next_1}, m_{next_2}\} \in M_{t'} \cup M_{t''}$ pojedynczo, aktualizując obecny stan M w następujący sposób (patrz rys. 3.5):

1. $\forall_{m \in M} : m_{next} \cap m = \emptyset$, do dopasowania M dodajemy m_{next}.

2. $\exists_{m=\{m_1,m_2\} \in M} : |m_{next} \cap m| = 1 \wedge \forall_{m' \in M, m' \neq m} : m_{next} \cap m' = \emptyset$ - bez straty ogólności załóżmy, że $m_1 = m_{next_1}$, czyli $m_{next} \cap m = \{m_1\}$, z M usuwamy m i dodajemy nowe dopasowanie $\{m_2, m_{next_2}\}$.

3. $\exists_{m=\{m_1,m_2\} \in M} : |m_{next} \cap m| = 1 \wedge \exists_{m'=\{m'_1,m'_2\} \in M, m \neq m'} : |m_{next} \cap m'| = 1$ - bez straty ogólności załóżmy, że $m_{next_1} = m_1$ i $m_{next_2} = m'_1$, usuwamy m i m' z M oraz dodajemy nowe dopasowanie $\{m_2, m'_2\}$.



Rysunek 3.5: Dodawanie nowej krawędzi m_{next} do dopasowania M.

4. $\exists_{m \in M} : |m_{next} \cap m| = 2$ - `CYKL`, $M_{t'}$ i $M_{t''}$ nie utworzą nam poprawnego rozwiązania.

Końcowy wynik dla parametrów $(t, \mathcal{D}, M)$ jest alternatywą po wszystkich $(t', \mathcal{D}_{t'}, M_{t'})$ i $(t'', \mathcal{D}_{t''}, M_{t''})$, dla których graf $(X_t, M_{t'} \cup M_{t''})$ jest acykliczny.

$$c[t][\mathcal{D}][M] = \bigvee_{\mathcal{D}_{t'}, \mathcal{D}_{t''}, M_{t'}, M_{t''}} c[t'][\mathcal{D}_{t'}][M_{t'}] \wedge c[t''][\mathcal{D}_{t''}][M_{t''}]$$

Dla każdego węzła mamy nie więcej niż $3^k \cdot k^k$ stanów. Zatem złożoność czasowa standardowego algorytmu dynamicznego po dekompozycji drzewowej dla problemu istnienia cyklu Hamiltona wynosi $k^{O(k)} \cdot n$, gdzie n jest liczbą wierzchołków grafu G danego na wejściu.

Rozdział 4

Algorytmy dynamiczne z zastosowaniem techniki Cut & Count

W poprzednim rozdziale zostały przedstawione klasyczne algorytmy dynamiczne po dekompozycji drzewowej dla dwóch problemów decyzyjnych: drzewa Steinera oraz cyklu Hamiltona. Złożoność czasowa obu tych algorytmów jest uwarunkowana liczbą wszystkich możliwych podziałów zbioru k-elementowego oraz liczbą dopasowań wierzchołków ze zbioru k-elementowego, która w obu przypadkach jest rzędu $k^{O(k)}$. W tym rodziale przedstawię randomizowane algorytmy dynamiczne po dekompozycji drzewowej, bazujące na technice Cut & Count opisanej w *Parameterized Algorithms* [1]. Technika Cut & Count redukuje problemy decyzyjne do problemów zliczania wszystkich możliwych rozwiązań modulo 2, eliminując czynnik $k^{O(k)}$, tym samym znacznie poprawiając złożoność czasową w stosunku do klasycznych algorytmów dynamicznych. Cut & Count tymczasowo dopuszcza rozwiązania niespójne (las w przypadku drzewa Steinera, zbiór cykli w przypadku cyklu Hamiltona), które następnie wzajemnie się znoszą przy zliczaniu modulo 2.

Ogólny schemat problemu, który rozwiązujemy z użyciem Cut & Count przedstawia się następująco: Dla uniwersum U niech $\mathcal{S} \subset 2^U$ oznacza zbiór szukanych rozwiązań. Pytamy, czy $\mathcal{S}$ jest pusty.

Technika Cut & Count bazuje na dwóch operacjach:

Cut - poluzuj wymagania dotyczące spójności szukanego rozwiązania, tj. na tym etapie dopuszczamy rozwiązania niespójne należące do zbioru $\mathcal{R} \supseteq \mathcal{S}$. Ponadto rozważamy zbiór $\mathcal{C}$ składający się z par (X, C), gdzie $X \in \mathcal{R}$ a C jest partycją podzbioru wierzchołków grafu wejściowego (V^1, V^2), z którą X jest kompatybilny (żadna spójna składowa nie ma niepustego przecięcia zarówno z V_1, jak i V_2).

Count - wyizoluj jedno z możliwych rozwiązań (o ile takie istnieje) poprzez przypisanie losowych wag elementom z uniwersum U. Rozbij zbiór $\mathcal{C}$ ze względu na wagi $w = \mathbf{w}(X)$, a następnie oblicz parzystości zbiorów $|\mathcal{C}_w|$ używając formuł rekurencyjnych. To pozwoli odrzucić wszystkie niepoprawne rozwiązania (niespójne), tj. $X \in \mathcal{R} \setminus \mathcal{S}$, ponieważ każde niespójne rozwiązanie jest kompatybilne z parzystą ilością partycji. Istnienie spójnego, wyizolowanego rozwiązania sprawi, że jedna z wartości $|\mathcal{C}_w|$ będzie równa 1.

4.1 Drzewo Steinera

Twierdzenie 2. Istnieje algorytm Monte Carlo z jednostronnym błędem - algorytm może zwrócić odpowiedź negatywną, kiedy rozwiązanie istnieje - który rozwiązuje problem drzewa Steinera w czasie $3^k \cdot n^{O(1)}$ mając na wejściu daną dekompozycję drzewową grafu o szerokości drzewowej k.

Dowód. Jak już zostało wspomniane, sprowadzamy problem decyzyjny do problemu parzystości liczby rozwiązań. Jednakże nie liczymy jej bezpośrednio, a uwzględniając rozwiązania niespójne - jak zostanie to pokazane, każde z nich wliczamy do końcowego wyniku parzystą liczbę razy, dzięki czemu wynik końcowy jest od nich niezależny.

Powołując się na *Parametrized Algorithms* [1], zdefiniujmy dwa zbiory $\mathcal{R}$ i $\mathcal{S}$. $\mathcal{R}$ niech będzie zbiorem „lasów Steinera", tj. zbiorem acyklicznych podgrafów G, których suma ma co najwyżej ℓ krawędzi i zawiera wszystie terminale K. $\mathcal{S}$ niech zawiera te podgrafy z $\mathcal{R}$, które są dodatkowo spójne:

$$\mathcal{R} = \{H \subseteq G : |E(H)| \leq \ell, K \subseteq V(H)\}$$
$$\mathcal{S} = \{H \in \mathcal{R} : H \textit{ jest spójny}\}$$

Dążymy do tego, by każdy element z $\mathcal{R} \setminus \mathcal{S}$ został zliczony parzystą liczbę razy, natomiast każdy element ze zbioru $\mathcal{S}$ nieparzystą liczbę razy. W tym celu definiujemy rodzinę podziałów zbioru $V(H)$ na dwa podzbiory V^1 i V^2:

$$\texttt{cuts}(V(H)) \coloneqq \{(V^1, V^2) : V^1 \cup V^2 = V(H) \wedge V^1 \cap V^2 = \emptyset\}$$

Definicja 8. *Graf H jest kompatybilny z podziałem* (V^1, V^2), jeśli $E(H) \subseteq \binom{V^1}{2} \cup \binom{V^2}{2}$, gdzie $\binom{X}{2}$ oznacza wszystkie dwuelementowe podzbiory zbioru X.

Zastanówmy się, jak wiele podziałów jest komptybilnych z grafem $H \in \mathcal{R}$. Skoro żadna z krawędzi nie może być rozpięta pomiędzy V^1 i V^2, każdy spójny komponent H musi być w całości w V^1 lub w V^2. Z powyższego dostajemy, że dla danego H mamy 2^c kompatybilnych podziałów, gdzie c jest liczbą spójnych składowych H. Każde niespójne H (z więcej niż jedną spójną składową) jest kompatybilne z parzystą liczbą podziałów, dzięki czemu rozwiązania niespójne się znoszą. Niestety spójne rozwiązania są również kompatybilne z parzystą liczbą podziałów - z dokładnie dwoma. Aby każde spójne rozwiązanie zostało zliczone dokładnie raz, wybieramy dowolny wierzchołek $v_0 \in K$ i przypisujemy go na stałe tylko do V^1. Zapiszmy formalnie nową definicję rodziny podziałów $V(H)$:

$$\texttt{cuts}(V(H), v_0) \coloneqq \{(V^1, V^2) : V^1 \cup V^2 = V(H) \wedge V^1 \cap V^2 = \emptyset \wedge v_0 \notin V^2\}$$

Pokażę teraz, że zamiast liczyć parzystość $|\mathcal{S}|$, możemy policzyć parzystość zbioru $\mathcal{C}$ zdefiniowanego następująco:

$$\mathcal{C} = \{(H, (V^1, V^2)) \in \mathcal{R} \times \texttt{cuts}(V(H), v_0) : H \textit{ jest kompatybilny z } (V^1, V^2)\}$$

Lemat 1. *Parzystość $|\mathcal{C}|$ jest równa parzytości $|\mathcal{S}|$.*

Dowód. Rozważmy graf H należący do $\mathcal{R}$, który ma c spójnych składowych. Wiezrchołki każdej spójnej składowej muszą się znaleźć w całości w V^1 albo w V^2. Jednakże spójna składowa zawierająca v_0 może się znaleźć tylko po stronie V^1. Z tego powodu H jest kompatybilny z 2^{c-1} podziałami. Dla $H \in \mathcal{S}$ liczba ta jest nieparzysta, natomiast dla $H \in \mathcal{R} \setminus \mathcal{S}$ parzysta. □

Wobec powyższego lematu, pozostaje pokazać algorytm obliczania parzystości $\mathcal{C}$. Dla każdego węzła $t \in V(T)$, liczby naturalnej $i \leq l$ i funkcji $f : X_t \to \{0, 1, 2\}$ obliczamy $c(t, f, i)$, które równa się liczbie par $(H, (V^1, V^2))$, takich że:

(i) H jest podgrafem (V_t, E_t) o dokładnie i krawędziach oraz H zawiera wszystkie terminale wprowadzone w bieżącym poddrzewie, tj. $K \cap V_t$.

(ii) (V^1, V^2) jest podziałem kompatybilnym z H, tj. $(H, (V^1, V^2)) \in \mathcal{C}$.

(iii) Przecięcie H z wierzchołkami należącymi do węzła t jest równe $V(H) \cap X_t = f^{-1}(1) \cup f^{-1}(2)$.

(iv) Funkcja f opisuje przynależność wierzchołków ze zbioru $X_t \setminus f^{-1}(0)$ do V^1 i V^2, tj. $V^j \cap V(H) \cap X_t = f^{-1}(j)$ gdzie $j \in \{1, 2\}$.

Zanim zdefiniuję rekurencyjne formuły obliczania wartości $c(t, f, i)$, pokażę w jaki sposób problem parzystości $\mathcal{C}$ redukuje się do problemu istnienia drzewa Steinera. Oczywistym jest, że może istnieć parzyście wiele różnych drzew Steinera o tej samej liczbie krawędzi, które się wzajemnie zniosą. Problem ten rozwiążemy poprzez wprowadzenie wag na krawędziach grafu G, które pozwolą nam rozróżniać poszczególne rozwiązania. Dla odpowiednio dużego N, losowo (niezależnie i z równym prawdopodobieństwem) wybieramy wagi ze zbioru $\{1, 2, \ldots, N\}$ i przyporządkowywujemy je krawędziom należącym do $E(G)$: $\mathbf{w}(e) \in \{1, 2, \ldots, N\}$.

Definicja 9. *Waga grafu H* jest sumą wag wszystkich jego krawędzi:

$$\mathbf{w}(H) = \sum_{e \in E(H)} \mathbf{w}(e)$$

Dla liczby naturalnej w, niech $\mathcal{R}_w = \{H \in \mathcal{R} : \mathbf{w}(H) = w\}$. Podobnie definiujemy $\mathcal{S}_w$ i $\mathcal{C}_w$. Z oczywistych względów $|\mathcal{S}_w| \mod 2 = |\mathcal{C}_w| \mod 2$ oraz zachodzi poniższa implikacja:

$$\exists_w : |\mathcal{S}_w| \mod 2 = 1 \implies \text{istnieje drzewo Steinera o } i \leq l \text{ krawędziach.}$$

Intuicyjnie, wystarczająco duże N powinno porozrzucać poprawne rozwiązania do różnych $\mathcal{S}_w$. Z tego wynika, że jeśli istnieje drzewo Steinera o $i \leq l$ krawędziach, to z wysokim prawdopodobieństwem $\exists_w : |\mathcal{S}_w| \mod 2 = 1$. Autorzy udowadniają za pomocą tzw. lematu izolującego coś znacznie silniejszego - mianowicie wystarczy wziąć $N = \Omega(|E(G)|)$, żeby ze stałym prawdopodobieństwem otrzymać co najmniej jeden zbiór $\mathcal{S}_w$ rozmiaru dokładnie 1.

Naszym uniwersum są krawędzie grafu wejściowego $U = E(G)$. Wagi na krawędziach losujemy ze zbioru $\{1, 2, \ldots, 2|U|\}$, a następnie liczymy parzystości $|\mathcal{C}_w|$. Modyfikujemy algorytm, dodając paramter w. Algorytm oblicza wartość $c(t, f, i, w)$ równą liczbie podgrafów H o wadze w, i krawędziach, kompatybilnych z podziałem (V_1, V_2). Poniżej przedstawiam rekurencyjne formuły zależne od typu węzła t. Rozwiązanie końcowe zależy od tego, czy istnieje $i \leq l$ oraz $w \leq 2|E|l$, dla których $c[r][0][i][w] \mod 2 = 1$. Jeśli tak, to istnieje drzewo

Steinera o i krawędziach. Jeśli nie, to z prawdopodobieństwem co najmniej $\frac{1}{2}$ nie istnieje drzewo Steinera o co najwyżej l krawędziach.

t jest `LIŚCIEM`

$$c[t][\emptyset][0][0] = 1$$

t jest węzłem `WPROWADZAJĄCYM` v - zauważmy, że $v \in K \implies f(v) \neq 0$ oraz $v = v_0 \implies f(v) = 1$. Jeśli któryś z wymienionych warunków nie jest spełniony $c[t][f][i][w] = 0$, wpp.

$$c(t, f, i, w) = c(t', f_{\big| X_{t'}}, i, w)$$

t jest węzłem `ZAPOMINAJĄCYM` v - dla węzła zapominającego sumujemy wyniki z węzła t' po różnych wartościach funkcji $f(v)$: 0, 1, 2.

$$\begin{aligned} c(t, f, i, w) = \; & c(t', f \cup \{v, 0\}, i, w) \\ & + c(t', f \cup \{v, 1\}, i, w) \\ & + c(t', f \cup \{v, 2\}, i, w) \end{aligned}$$

t jest węzłem `WPROWADZAJĄCYM KRAWĘDŹ` uv - zauważmy, że krawędź uv może zostać dodana do rozwiązania, kiedy parametry (t, f, i, w) opisują stan, w którym $f(u) = f(v) \neq 0$ - uv jest zgodna z podziałem (V^1, V^2) oraz u i v nie są wierzchołkami izolowanymi.

$$c(t, f, i, w) = \begin{cases} c(t', f, i, w) + c(t', f, i-1, w - w_{uv}), \text{ jeśli } f(u) = f(v) \neq 0 \\ c(t', f, i, w), \text{ wpp.} \end{cases}$$

t jest węzłem `SCALAJĄCYM` - dla węzła scalającego wynik jest sumą po wszystkich parach $(\mathcal{C}_{\mathbf{w}(H')}, \mathcal{C}_{\mathbf{w}(H'')})$, gdzie H' jest podgrafem $G_{t'}$, a H'' podgrafem $G_{t''}$.

$$c(t, f, i, w) = \sum_{\substack{i'+i''=i \\ w'+w''=w}} c(t', f, i', w') + c(t'', f, i'', w'')$$

Złożoność powyższego algorytmu wynosi $3^k \cdot n^{O(1)}$, czyli zależność złożoności czasowej od szerokości drzewowej jest rzędu $2^{O(k)}$, a nie jak w przypadku standardowego algorytmu dynamicznego $k^{O(k)}$, co kończy dowód twierdzenia 2. □

4.2 Cykl Hamiltona

W tym rozdziale przedstawię zastosowanie techniki Cut & Count do rozwiązania problemu istnienia cyklu Hamiltona. Ponieważ aspekty techniczne w większości pokrywają się z tym, co zostało przedstawione w poprzednim paragrafie dla drzewa Steinera, skupię się wyłącznie na niuansach charakterystycznych dla cyklu Hamiltona.

Pominięcie wymogu spójności redukuje problem cyklu Hamiltona do problemu pokrycia wierzchołkowego grafu k cyklami, gdzie w naszym przypadku $k \in \{1, \ldots, \left\lfloor \frac{|V|}{2} \right\rfloor\}$. Zbiór $\mathcal{R}$

jest zbiorem pokryć H, które składają się z cykli co najmniej dwuelementowych i których suma pokrywa wszystkie wierzchołki grafu G:

$$\mathcal{R} = \{H \subseteq G : V(H) = V(G), \forall u \in V(H) : deg(u) = 2\}$$
$$\mathcal{S} = \{H \in \mathcal{R} : H \textit{ jest spójny}\}$$

Zauważmy, że o ile w przypadku problemu drzewa Steinera mamy zachowany niezmiennik, że jego ślad składa się z drzew, o tyle w przypadku cyklu Hamiltona podobny niezmiennik nie zachodzi. Rozwiązanie dla bieżącego podgrafu G_t składa się nie tylko z cykli, ale także ze ścieżek. Zliczamy takie rozwiązania, dopóki istnieje możliwość domknięcia wszystkich ścieżek (tzn. w węźle zapominającym v wymagamy, by $deg(v) = 2$).

Zbiór $\mathcal{C}$ definiujemy podobnie jak dla drzewa Steinera, z dwiema drobnymi modyfikacjami. Jedna z nich dotyczy v_0, który dla cyklu Hamiltona jest dowolnym, ale ustalonym wierzchołkiem ze zbioru $V(G)$. Druga natomiast dotyczy podziału (V^1, V^2), który definiujemy tylko na wierzchołkach o stopniu 1.

$$\texttt{cuts}(V_{deg_1}(H), v_0) \coloneqq \{(V^1, V^2) : V^1 \cup V^2 = V_{deg_1}(H) \wedge V^1 \cap V^2 = \emptyset \wedge v_0 \notin V^2\}$$

$$\mathcal{C} = \{(H, (V^1, V^2)) \in \mathcal{R} \times \texttt{cuts}(V_{deg_1}(H), v_0) : H \textit{ jest kompatybilny z } (V^1, V^2)\}$$

Zbiory $\mathcal{R}_w$, $\mathcal{S}_w$ oraz $\mathcal{C}_w$ definiujemy analogicznie jak dla problemu drzewa Steinera. Rysunek 4.2 przedstawia ślad cyklu Hamiltona w węźle t. Zbiór X_t jest podzielony na cztery podzbiory składające się z wierzchołków o różnej charakterystyce:

X_{t_0} jest zbiorem wierzchołków o stopniu 0.
$X_{t_{1_1}}$ jest zbiorem wierzchołków o stopniu 1, należacych do V^1.
$X_{t_{1_2}}$ jest zbiorem wierzchołków o stopniu 1, należących do V^2.
X_{t_2} jest zbiorem wierzchołków o stopniu 2.

Dla każdego węzła $t \in V(T)$, funkcji $f : X_t \to \{0, 1_1, 1_2, 2\}$ oraz wagi w rekurencyjnie obliczamy $c(t, f, w)$ będące liczbą par $(H, (V^1, V^2))$, takich że:

(i) H jest podgrafem (V_t, E_t) oraz H zawiera wszystkie wierzchołki dotychczasowo wprowadzone w bieżącym poddrzewie, tj. $V(H) = V_t$.

(ii) (V^1, V^2) jest podziałem kompatybilnym z H.

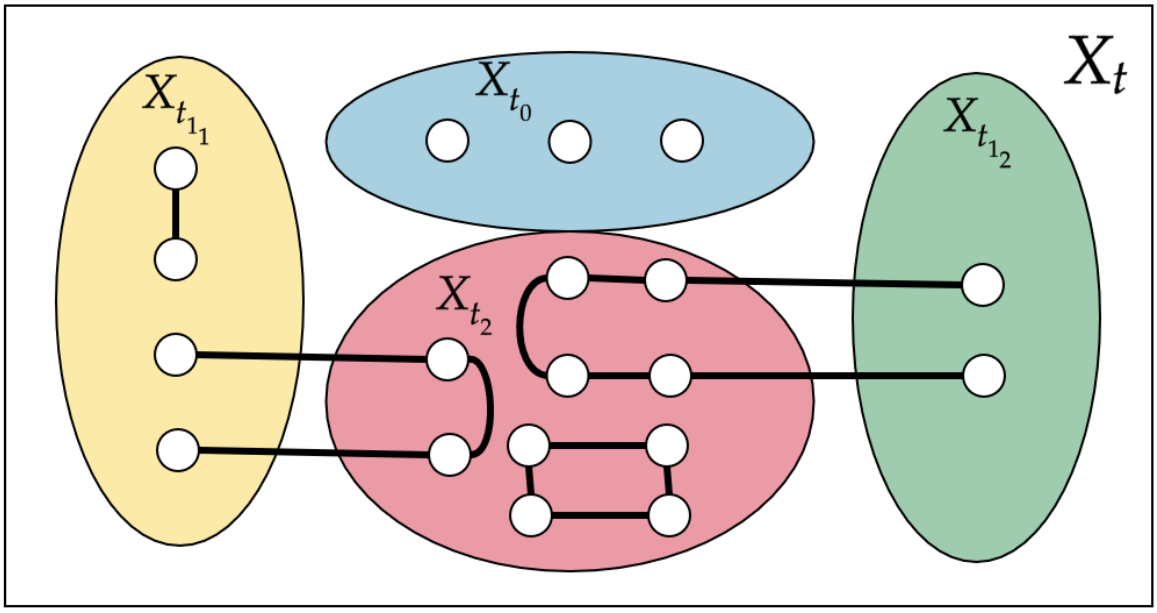
(iii) Funkcja f opisuje przynależność wierzchołków ze zbioru X_t do podzbiorów $X_{t_0}, X_{t_{1_1}}, X_{t_{1_2}}, X_{t_2}$, tj. $X_{t_j} \cap V(H) = f^{-1}(j)$, gdzie $j \in \{0, 1_1, 1_2, 2\}$.

(iv) $X_t = f^{-1}(0) \cup f^{-1}(1_1) \cup f^{-1}(1_2) \cup f^{-1}(2)$

(v) $V^1 \cup V^2$ nie ma wierzchołków poza X_t, innymi słowy $V_{deg_1}(H) \cap X_t = V_{deg_1}(H)$, inaczej nieodwołalnie zostawilibyśmy niedomknięty cykl.

(vi) $X_{t_{1_1}} = V^1$ oraz $X_{t_{1_2}} = V^2$

(vii) $v \in X_{t_j} \implies deg(v) = j$ dla $j \in \{0, 1, 2\}$



Rysunek 4.1: X_t ze śladem cyklu Hamiltona.

(viii) $\sum_{e \in E(H)} \mathbf{w}(e) = w$

Zauważmy, że każdy wierzchołek musi po kolei należeć do $X_{t_0} \to \{X_{t_{1_1}}\ lub\ X_{t_{1_2}}\} \to X_{t_2}$. Poniżej przedstawiam definicje rekurencyjne obliczania wartości $c(t, f, w)$. Algorytm zwraca, że istnieje cykl Hamiltona, jeśli $\exists_{w \leq nN} : c(r, \emptyset, w) \mod 2 = 1$, gdzie r jest korzeniem dekompozycji drzewowej, a $n = |V(G)|$.

t jest `LIŚCIEM`

$$c(t, \emptyset, 0) = 1$$

t jest węzłem `WPROWADZAJĄCYM` v - wierzchołek v w momencie wprowadzania nie ma incydentnej krawędzi, zatem jeśli $f(v) \neq 0$, to $c(t, f, w) = 0$, wpp.

$$c(t, f, w) = c(t', f_{\big|_{X_{t'}}}, w)$$

t jest węzłem `ZAPOMINAJĄCYM` v - wierzchołek może zostać zapomniany wtw. gdy znajduje się w X_{t_2}, czyli ma dwie incydentne krawędzie, jeśli $f(v) \neq 2$, to $c(t, f, w) = 0$, wpp.

$$c(t, f, w) = c(t', f \cup \{(v, 2)\}, w)$$

t jest węzłem `WPROWADZAJĄCYM KRAWĘDŹ` uv - następujące warunki muszą być spełnione, żeby można było dodać krawędź uv:

	$X_{t'_0}$	$X_{t'_{1_1}}$	$X_{t'_{1_2}}$	$X_{t'_2}$
$X_{t'_0}$	$(1_1, 1_1) \vee (1_2, 1_2)$	$(1_1, 2)$	$\emptyset$	$(1_2, 2)$
$X_{t'_{1_1}}$	$(2, 1_1)$	$(2, 2)$	$\emptyset$	$\emptyset$
$X_{t'_{1_2}}$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$
$X_{t'_2}$	$(2, 1_2)$	$\emptyset$	$\emptyset$	$(2, 2)$

Tablica 4.1: Tabela przedstawia, jak zmienia się przynależność wierzchołków u i v do zbiorów X_{t_i} po połączeniu ich krawędzią. Tabelę należy rozumieć w następujący sposób: $tab[f'(u)][f'(v)] = (f(u), f(v))$, i jest tożsame z X_{t_i}, a f' odpowiada funkcji f w węźle t'.

	$X_{t''_0}$	$X_{t''_{1_1}}$	$X_{t''_{1_2}}$	$X_{t''_2}$
$X_{t'_0}$	0	1_1	2	1_2
$X_{t'_{1_1}}$	1_1	2	$-$	$-$
$X_{t'_{1_2}}$	2	$-$	$-$	$-$
$X_{t'_2}$	1_2	$-$	$-$	2

Tablica 4.2: Zmiana przynależności wierzchołka do zbioru X_{t_i} po scaleniu dwóch podgrafów. Tabelę należy interpretować następująco: $tab[f'(v)][f''(v)] = f(v)$, i jest tożsame z X_{t_i}, f' i f'' odpowiadają funkcji f w węzłach t' i t''.

(i) $deg(u) < 2$ oraz $deg(v) < 2$

(ii) uv jest kompatybilna z podziałem (V^1, V^2).

(iii) $tab[f'(u)][f'(v)] \neq -$, gdzie tab odnosi się do tabeli 4.1.

$$c(t, f, w) = \begin{cases} c(t', f, w) + c(t', f', w - w_{uv}), & \text{jeśli spełnione są warunki (i), (ii), (iii)} \\ c(t', f, w), & \text{wpp.} \end{cases}$$

t jest węzłem `SCALAJĄCYM` - żeby otrzymać poprawne rozwiązanie w wyniku scalenia, muszą zachodzić następujące warunki:

(i) $\forall v \in X_t$: $deg_{t'}(v) + deg_{t''}(v) \leq 2$

(ii) $\forall v \in X_t$: jeśli $v \in X_{t'_{1_1}}$, to $v \notin X_{t''_{1_2}}$

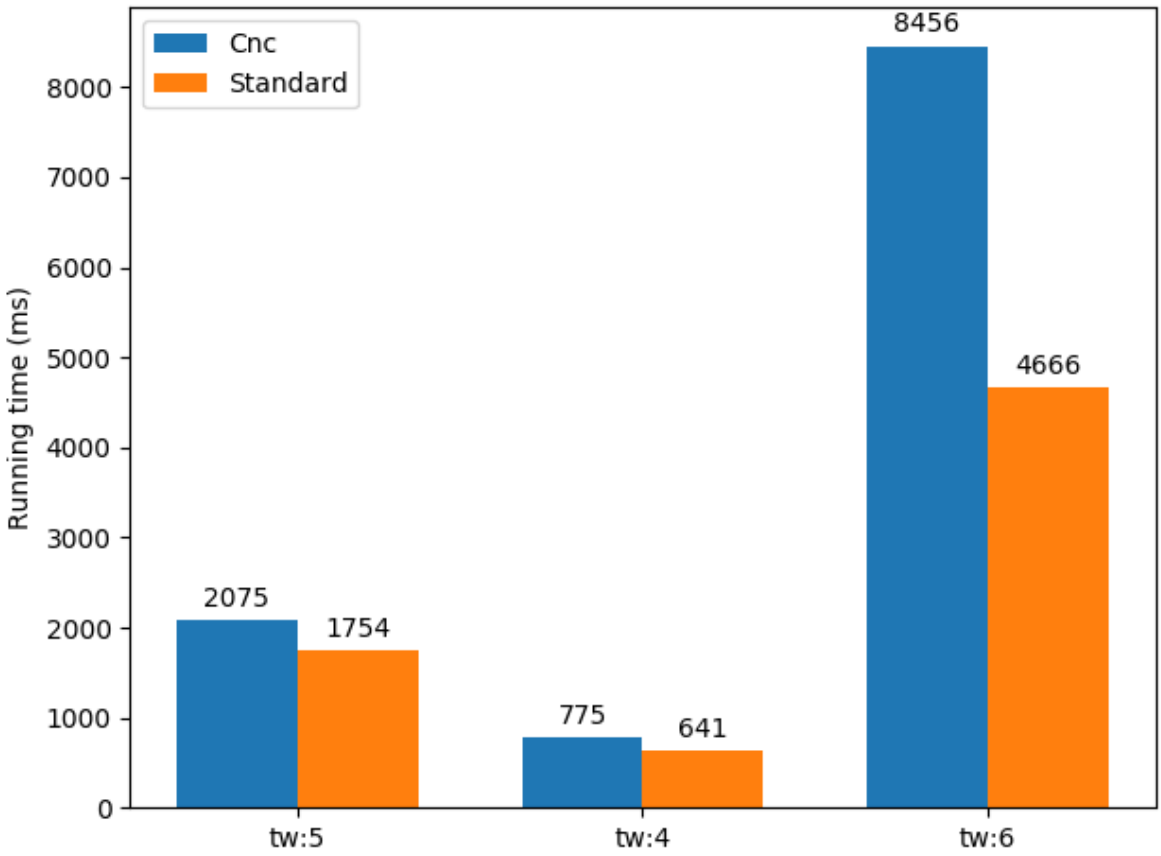
(iii) $tab[f'(v)][f''(v)] \neq -$, gdzie tab odnosi się do tabeli 4.2.

$$c(t, f, w) = \begin{cases} \sum_{w'+w''=w} c(t', f', w') + c(t'', f'', w''), & \text{jeśli spełnione są warunki (i), (ii), (iii)} \\ 0, & \text{wpp.} \end{cases}$$

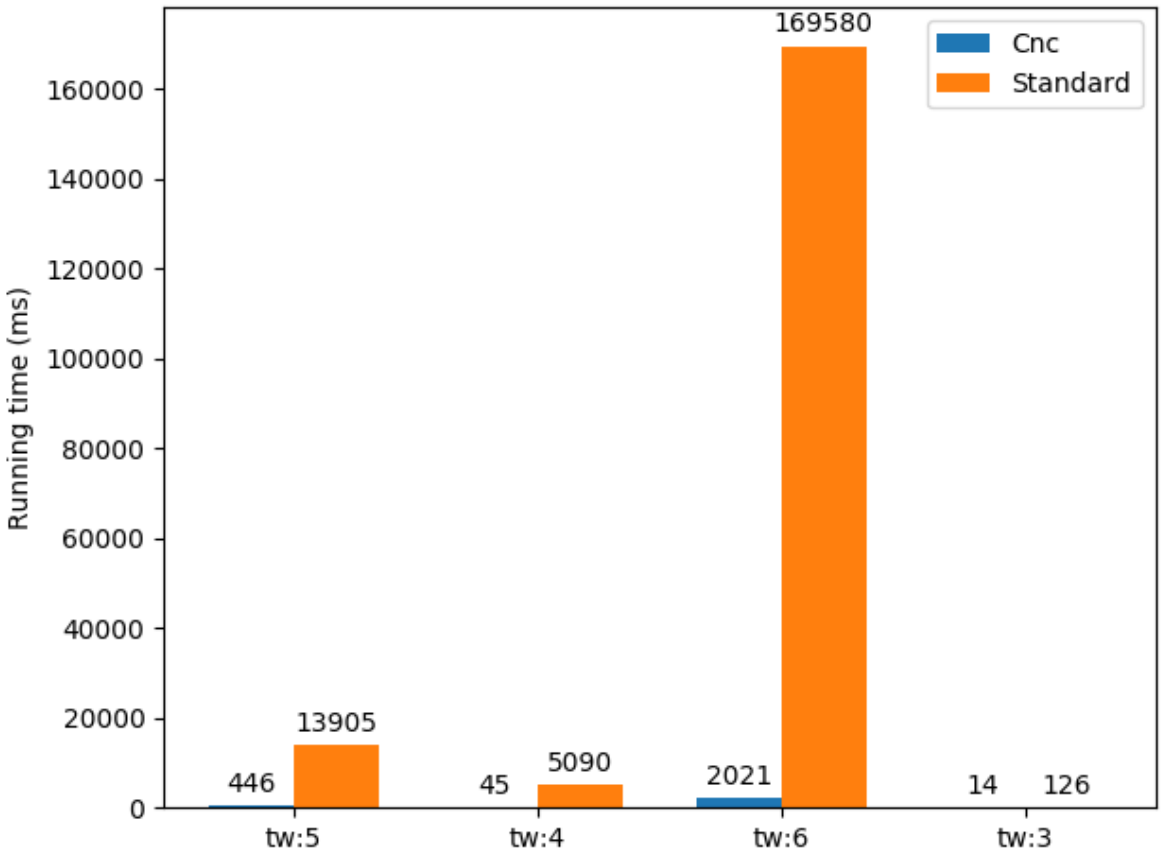
Złożoność powyższego algorytmu wynosi $4^k \cdot n^{O(1)}$, czyli zależność złożoności czasowej od szerokości drzewowej jest rzędu $2^{O(k)}$, a nie jak w przypadku standardowego algorytmu dynamicznego $k^{O(k)}$.

Rozdział 5

Porównanie wydajności zaimplementowanych algorytmów



Rysunek 5.1: Porównanie czas działania algorytmów dla problemu cyklu Hamiltona.



Rysunek 5.2: Porównanie czas działania algorytmów dla problemu drzewa Steinera

Bibliografia

[1] M. Cygan, F. V. Fomin, Ł. Kowalik, D. Lokshtanov, D. Marx, M. Pilipczuk, M. Pilipczuk, S. Saurabh *Parameterized Algorithms*.

[2] H. L. Bodlaender. *A linear-time algorithm for finding tree-decompositions of small treewidth*. SIAM J. Comput. 25:6 (1996) 1305-1317

[3] T. Kloks. *Treewidth. Computations and approximations*. Lecture Notes in Computer Science, 842, 1994.

[4] P. Kyzioł: Cut & Count - Code repository, 2019
`https://github.com/polapl/Cut-and-Count`