Szczególne przypadki problemów NP-trudnych

szerokość drzewowa

Janusz Parfieniuk

Jak rozwiązywać problemy NP-trudne?

- **algorytmy aproksymacyjne** – *czas wielomianowy ale otrzymane rozwiązanie może odbiegać od najlepszego*

- **dokładnie** - *wiemy, że w najgorszym przypadku nie będziemy w stanie rozwiązać problemu w czasie wielomianowym.*

Co się dzieje jeżeli mamy szczególny przypadek o określonej strukturze?

Przykład I – znajdowanie małego pokrycia wierzchołkowego

Problem:

Mając dany graf G=(V,E) oraz liczbę całkowitą k, znaleźć pokrycie wierzchołkowe rozmiaru k.

(pokrycie wierzchołkowe to taki podzbiór zbioru wierzchołków, że każda krawędź z grafu ma przynajmniej jeden koniec w tym zbiorze)

Rozwiązanie podstawowe:

Dla każdego podzbioru zbioru wierzchołków o rozmiarze k sprawdź czy jest on pokryciem wierzchołkowym.

Złożoność algorytmu:

- ilość podzbiorów rozmiaru k $O\left(\binom{n}{k}\right)$

- sprawdzenie czy zbiór jest pokryciem wierzchołkowym: $O(nk)$

- sumaryczna złożoność $O\left(nk\binom{n}{k}\right)=O(kn^{k+1})$

Rozwiązanie uwzględniające to, że k jest „małe”

Weź dowolną krawędź (u,v) z grafu G. Sprawdź rekurencyjnie czy graf G-{u} lub G-{v} zawiera k-1 elementowe pokrycie wierzchołkowe.

Złożoność:

- koszt pojedynczego wywołania procedury: $O(kn)$

- liczba wywołań procedury: $O(2^k)$

- sumaryczna złożoność: $O(2^k kn)$

Rozwiązywanie NP-trudnych problemów grafowych dla drzew

Dla wielu NP-trudnych problemów grafowych wykazano, że w przypadku, gdy graf wejściowy jest drzewem, da się je szybko rozwiązać.

Zamiast narzucać ograniczenia na rozmiar danych wejściowych, wymaga się na wejściu parametrów o określonej strukturze.

Przykład II – zachłanny algorytm dla znajdowania zbioru niezależnego w drzewie

Problem:

Mając dany graf G=(V,E) znaleźć najliczniejszy zbiór niezależny (dla grafu z wagami – zbiór niezależny o największej wadze).

(zbiór niezależny jest to podzbiór S zbioru wierzchołków grafu G taki ,że pomiędzy żadnymi dwoma wierzchołkami z S nie ma krawędzi w G)

Własność

Jeżeli T=(V,E) jest drzewem oraz v jest liściem tego drzewa, wówczas istnieje maksymalny zbiór niezależny zawierający v.

Dowód:
Rozważmy maksymalny zbiór niezależny S, niech e=(u,v) będzie jedyną krawędzią incydentną z v. S jest maksymalny a więc albo u albo v należą do S (jeżeli żaden by nie należał wówczas można by było dodać v do S zwiększając tym samym jego rozmiar – sprzeczność). Jeżeli v jest w zbiorze S własność zachodzi. Jeżeli v nie należy do S wówczas u należy do S i możemy uzyskać inny zbiór niezależny S` zamieniając w S u przez v.

Algorytm zachłanny dla lasów

```
S = {}
while (E <> {}) do
    v = liść z lasu drzew G
    e = (u,v) // jedyna krawędź incydentna z v
    S = S + {v}
    G = G – {v, u}
done
S = S + V // wierzchołki pozostałe w V to drzewa o jednym
          // wierzchołku
```

Złożoność algorytmu:

- znajdowanie wierzchołka v (łącznie z usunięciem wchodzącej do niego krawędzi): $O(1)$

- sumaryczna złożoność: $O(n)$

Algorytm dla lasów z wagami

for (u wierzchołki z G w kolejności postorder) do
if (u jest liściem)
Mout[u] = 0
Min[u] = w(u)
else
Mout[u] = $\sum_{v \in children(u)} max(Mout[u], Min[u])$

Min[u] = w(u) + $\sum_{v \in children(u)} Mout[u]$
endif
done

Złożoność algorytmu:

- liczba operacji wykonywanych dla danego wierzchołka: $O(1)$

- złożoność sumaryczna: $O(n)$

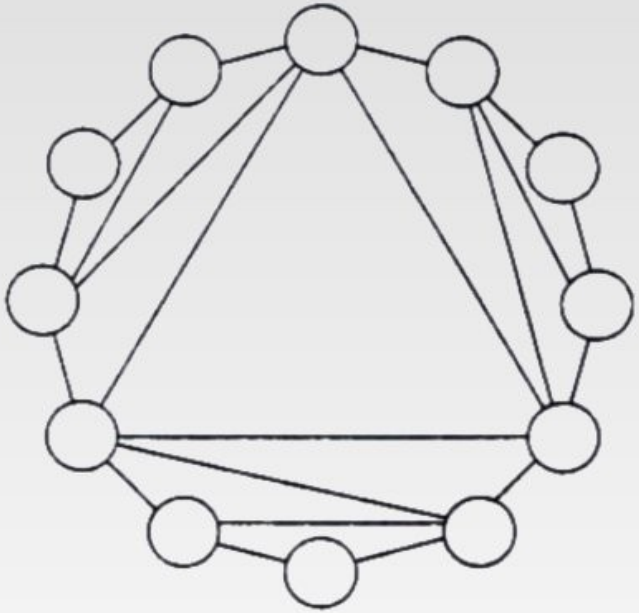
Drzewowa dekompozycja grafu

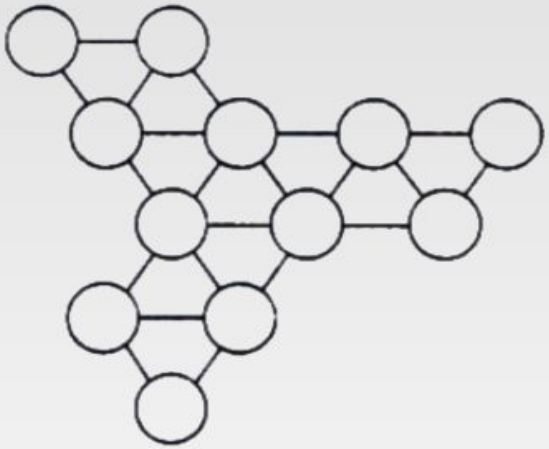
Po co wykonuje się dekompozycję do drzew?

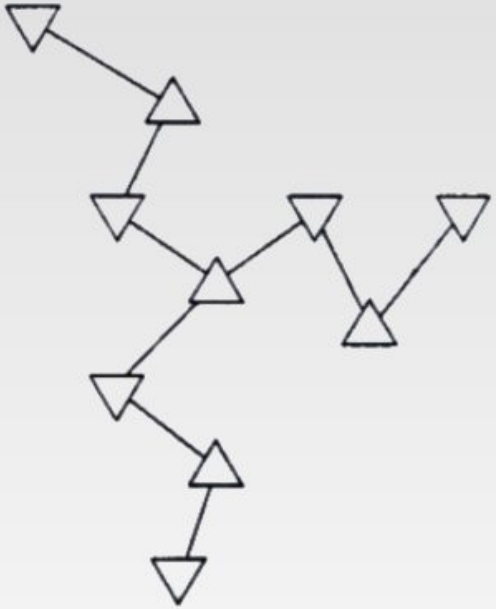
- drzewa łatwo się rozdziela na rozłączne elementy (podproblemy) dzięki temu można rozwiązywać problemy mniejszych rozmiarów i z nich wyliczać ostateczne rozwiązanie

(programowanie dynamiczne)

Drzewowa dekompozycja grafu







Dekompozycja drzewowa grafu:

Definicja:

Dekompozycja drzewowa grafu G=(V,E) jest to drzewo T=(V`,E`) takie, że wierzchołkami drzewa są podzbiory grafu G (dla wierzchołka drzewa t zbiór ten będziemy oznaczać V_t) oraz spełnione są następujące własności:

1. Każdy wierzchołek grafu G należy do co najmniej jednego wierzchołka drzewa T (pokrycie wierzchołkowe)

2. Dla każdej krawędzi grafu G istnieje wierzchołek drzewa zawierający oba końce krawędzi (pokrycie krawędziowe)

3. Jeżeli dwa wierzchołki drzewa x i y zawierają wierzchołek grafu v to każdy wierzchołek drzewa na ścieżce prostej z x do y także zawiera wierzchołek v. (spójność)

Własności dekompozycji drzewowej:

Notacja:
Niech T` będzie podgrafem T, wówczas $G_{T`}$ oznacza podgraf G indukowany przez zbiór będący sumą wierzchołków z T` (czyli sumą podzbiorów wierzchołków z G).

Własność I:
Niech las powstały w wyniku usunięcia wierzchołka t z drzewa T (T-t) składa się z drzew $T_1,...,T_d$, wówczas podgrafy:

$$G_{T1}\text{-}V_t,\ G_{T2}\text{-}V_t,...,G_{Td}\text{-}V_t$$

nie mają wspólnych wierzchołków i nie ma pomiędzy nimi żadnych krawędzi.

Dowód:

a) podgrafy G_{ti}-V_t nie mają węzłów wspólnych:
Jeżeli istniałby taki węzeł v, należący do jakiegoś G_{ti}-V_t oraz do G_{tj}-V_t dla i<>j, wówczas v należałby do wierzchołka x drzewa T_i oraz wierzchołka y drzewa T_j. t leży na ścieżce prostej od x do y a więc (ze spójności) v musi należeć także do V_t, w związku z tym nie może należeć ani do G_{ti}-V_t ani do G_{tj}-V_t.

b) nie istnieje krawędź e=(u,v) w G taka, że u należy do G_{ti}-V_t a v należy do G_{tj}-V_t dla i<>j:
Z pokrycia krawędziowego wiemy, że {u,v} należy do V_x, jednocześnie x nie może jednocześnie być w poddrzewie T_i oraz T_j. Załóżmy, że x należy do T_j. Wierzchołek grafu u należy do jakiegoś V_y z T_i. Wierzchołek drzewa t leży na ścieżce prostej z x do y a więc skoro u należy do V_y oraz V_x, to u należy także do V_t a więc nie może należeć ani do G_{ti}-V_t ani do G_{tj}-V_t.

Drzewowa dekompozycja grafu – szerokość drzewowa, własność II

Własność II:

Niech X i Y będą dwoma drzewami powstałymi w wyniku usunięcia krawędzi (x,y) z T. Wówczas usunięcie ($V_x \cap V_y$) ze zbioru wierzchołków grafu rozspójnia G na dwa podgrafy $G_x - (V_x \cap V_y)$ oraz $G_y - (V_x \cap V_y)$.

Drzewowa dekompozycja grafu – szerokość drzewowa, własność II

Dowód (analogiczny do dowodu własności I):

a) podgrafy $G_x - (V_x \cap V_y)$ i $Gy - (V_x \cap V_y)$ nie mają wspólnych wierzchołków

Załóżmy, że istnieje wspólny wierzchołek $\underline{v}$ należący do G_x i G_y, wówczas $\underline{v}$ należy jednocześnie do V_x i V_y a więc nie może należeć ani do $G_x - (V_x \cap V_y)$ ani do $G_y - (V_x \cap V_y)$

b) nie istnieje krawędź $e=(u,v)$ łącząca $G_x - (V_x \cap V_y)$ i $G_y - (V_x \cap V_y)$

Jeżeli istniałaby taka krawędź $e=(u,v)$ wówczas, z pokrycia krawędziowego, $\{u,v\}$ należy do jakiegoś V_z. Załóżmy, że wierzchołek $\underline{z}$ należy do X. Wierzchołek grafu $\underline{v}$ należy do jakiegoś V_w z poddrzewa Y. W związku z tym $\underline{v}$ należy także do V_x i V_y a więc $\underline{v}$ należy do $(V_x \cap V_y)$ czyli nie może należeć do $Gy - (V_x \cap V_y)$.

Drzewowa dekompozycja grafu – szerokość drzewowa, definicja

Definicja:
Szerokość dekompozycji drzewowej T grafu G jest to:

$$\max_{po\ t \in T} |V_t| - 1$$

Definicja:
Szerokość drzewowa grafu G jest to minimalna szerokość dekompozycji drzewowej dla tego grafu.

Ciekawostka:
Spójny graf G ma szerokość drzewową 1 wtedy i tylko wtedy gdy G jest drzewem.

Drzewowa dekompozycja grafu – dekompozycja bez powtórzeń

Definicja:

Dekompozycja drzewowa T grafu G jest dekompozycją bez powtórzeń wtedy i tylko wtedy gdy dla żadnej krawędzi (x,y) z drzewa nie zachodzi $(V_x \subset V_y)$.

Algorytm:
(przekształcania dekompozycji grafowej do dekompozycji bez powtórzeń)

Jeżeli dla jakiejś krawędzi (x,y) z drzewa T zachodzi $(V_x \subset V_y)$ *wówczas sklejamy ze sobą wierzchołki x i y.*

Drzewowa dekompozycja grafu – dekompozycja bez powtórzeń

Własność III:
Każda dekompozycja drzewowa bez powtórzeń grafu o n wierzchołkach zawiera nie więcej niż n elementów.

Dowód:
Indukcja po n.
- n = 1: oczywiste
- n > 1:
Mając drzewową dekompozycję T n wierzchołkowego grafu G weźmy liść t drzewa T. T jest dekompozycją bez powtórzeń a więc V_t zawiera elementy nie należące do jego sąsiadów (i ze spójności nie należą także do żadnego innego wierzchołka). Niech U będzie zbiorem tych wszystkich elementów. Poprzez usunięcie t z drzewa T uzyskujemy dekompozycję drzewową bez powtórzeń dla grafu G-U. Uzyskane drzewo ma n-1 elementów co kończy dowód.

Problem zbioru niezależnego

Intuicja:

Optymalny zbiór niezależny ma z V_t pewne elementy wspólne, niech będzie to podzbiór U. Nie wiemy jednak, o który podzbiór U chodzi. Wyliczamy więc wszystkie możliwości dla podzbioru U. V_t może mieć maksymalnie rozmiar w+1 co daje 2^{w+1} możliwości (w wielu wypadkach jest to znacznie lepiej niż 2^n). Dzięki własnościom I i II w dwóch różnych poddrzewach T problemy można rozwiązywać niezależnie.

Drzewowa dekompozycja grafu – problem zbioru niezależnego

Definicja:

Niech podzbiór U zbioru V_t reprezentuje przecięcie optymalnego rozwiązania z V_t. Dla każdego niezaleznego zbioru ($U \subset V_t$) definiujemy:

$$f_t(U) = \max_{S \text{ zbiór niezależny w } Gt \text{ taki że } (S \cap Vt) = U} w(S)$$

$f_t(U)$ jest niezdefiniowany jeżeli U nie jest zbiorem niezależnym.

Drzewowa dekompozycja grafu – problem zbioru niezależnego

Algorytm:

T = drzewo dekompozycji grafu G bez powtórzeń
r = korzeń drzewa T
for (t – wierzchołki T w kolejności postorder) do
    if (t jest liściem)
        for (U – niezależne zbiory w V_t*)*
            $F_t(U) = w(U)$
        done
    else
        for (U – niezależne zbiory w V_t*)*
            $F_t(U)=w(U)+\sum_{i=1}^{d} Z_i$
        done
    endif
done

$$Z_i=max\{F_{t_i}(U_i)-w(U_i\cap U): U_i\cap V_t=U\cap V_{t_i}\, i\, U_i\subseteq V_{t_i}\, jest\, niezależny\}$$

Drzewowa dekompozycja grafu – problem zbioru niezależnego

Analiza algorytmu:

- sprawdzenie czy $U_i \cap V_t = U \cap V_{ti}$*:*	$O(w)$
- liczba zbiorów niezależnych w V_{ti}	$O(2^{w+1})$
- liczba generowanych zbiorów niezależnych w każdym węźle:	$O(2^{w+1})$
- liczba węzłów	$O(n)$
- sumaryczna złożoność	$O(4^{w+2}wn)$

Drzewowa dekompozycja grafu – konstrukcja

Znalezienie szerokości drzewowej danego grafu G jest problemem NP-trudnym.

Jeżeli interesujemy się grafami dla których szerokość drzewowa w jest niezbyt dużą stałą, wówczas istnieje algorytm, który dla zadanego grafu G o szerokości drzewowej mniejszej niż w zwróci dekompozycję drzewową G o szerokości mniejszej niż 4w.

(czas algorytmu działania O(f(w)mn) gdzie m to liczba krawędzi n liczba wierzchołków)

Drzewowa dekompozycja grafu – konstrukcja

Twierdzenie 1:
Niech G=(V,E) ma m krawędzi, X będzie podzbiorem jego wierzchołków o liczności k oraz w <= k będzie zadanym parametrem. Możemy sprawdzić czy X jest w-spójny w czasie O(f(k)m). Dodatkowo, jeżeli X nie jest w-spójny, możemy otrzymać dowód w postaci zbiorów (Y,Z ⊂ X) oraz (S ⊂ V) takich że |S|<|Y|=|Z|<= w i nie istnieje ścieżka z Y-S do Z-S w G-S.

Szkic dowodu/algorytmu:
Generujemy wszystkie możliwe podzbiory (2^k możliwości) i dla każdej pary Y,Z (takiej, że l = |Y|=|Z| <= w sprawdzamy czy istnieje między X i Z przepływ w grafie G o wartości co najmniej l (ścieżki muszą być wierzchołkowo różne więc wierzchołkom przypisujemy jednostkową przepustowość).

Drzewowa dekompozycja grafu – konstrukcja

Twierdzenie 2:
Jeżeli graf G zawiera (w+1)-spójny podzbiór wierzchołków X o rozmiarze co najmniej 3w, wówczas G ma szerokość drzewową przynajmniej w.

Szkic dowodu:
Dowód polega na pokazaniu przez sprzeczność, że jeżeli G zawiera (w+1)-spójny podzbiór wierzchołków X o rozmiarze co najmniej 3w i istnieje drzewo T będące dekompozycją drzewową tego grafu o szerokości mniejszej niż w (czyli każdy V_t zawiera maksymalnie w wierzchołków) wówczas można znaleźć wierzchołek, w którego poddrzewie jest co najmniej 2w wierzchołków i jest najbardziej oddalonym wierzchołkiem od korzenia o tej własności. Następnie należy pokazać, że usunięcie tego wierzchołka jest sprzeczne z założeniem o (w+1)-spójności grafu.

Drzewowa dekompozycja grafu – konstrukcja

Definicja:
Jeżeli C jest spójnym podgrafem grafu G-U to każdy wierzchołek (u ∈ U) jest sąsiadem C jeżeli istnieje wierzchołek (v ∈ C) oraz krawędź (v,u) ∈ E.

Niezmienniki algorytmu:
1. *W dowolnym momencie wykonania algorytmu, każdy spójny podgraf C grafu G-U ma co najwyżej 3w sąsiadów oraz istnieje* V_t *zawierająca wszystkich sąsiadów C.*
2. *W dowolnym momencie działania algorytmu skonstruowane drzewo jest częściowym drzewem dekompozycji o rozmiarze mniejszym niż 4w (czyli jeżeli drzewo pokrywa U wierzchołków to jest to drzewo dekompozycji dla podgrafu G złożonego z wierzchołków z U).*

Drzewowa dekompozycja grafu – konstrukcja

Algorytm:

1. Rozpoczynamy z pustym drzewem dekompozycji.

2. Dopóki G-U jest niepusty dodajemy nowy wierzchołek do zbudowanego częściowego drzewa dekompozycji:

T – drzewo częściowej dekompozycji (na zbiorze U wierzchołków)

C – spójna składowa grafu G-U

X – zbiór sąsiadów C

t – wierzchołek zawierający cały zbiór X

Możliwe są dwie sytuacje:

a) zbiór X ma mniej niż 3w elementów

b) zbiór X ma dokładnie 3w elementów

Drzewowa dekompozycja grafu – konstrukcja

Algorytm cd. 1:

a) zbiór X ma mniej niż 3w elementów

- bierzemy dowolny wierzchołek v z C i do drzewa T dodajemy nowy węzeł s = (X ∪ {v}) oraz krawędź (s,t)

operacja ta nie wpływa na niezmiennik:

1. |X| < 3w a więc |X ∪ {v}| <= 3w

2. ponieważ wszyscy sąsiedzi wierzchołka v są w zbiorze X, łatwo stwierdzić że nadal mamy drzewo częściowej dekompozycji.

Drzewowa dekompozycja grafu – konstrukcja

Algorytm cd. 2:

b) zbiór X ma dokładnie 3w elementów

1. Wykonujemy sprawdzenie czy X jest (w+1)-spójnym zbiorem (z Twierdzenia 1 sprawdzenie to wykonujemy w czasie O(f(w)m))
- jeżeli zbiór X okaże się być (w+1)-spójny, wówczas z Twierdzenia 2 wiemy, że jest graf G ma szerokość drzewową co najmniej w i kończymy algorytm
- jeżeli zbiór X nie jest (w+1)-spójny wówczas w wyniku wykonania algorytmu z Twierdzenia 1 dostaniemy zbiory (Y,Z ⊂ X) oraz (S ⊂ V) takie że |S| < |Y|=|Z| <= w+1 i nie istnieje ścieżka z Y-S do Z-S w G-S. Zbiory te posłużą nam do rozszerzenia drzewa T o kolejny węzeł.

2. Niech S` składa się z węzłów które należą do S ∩ (Y ∪ Z ∪ C). S`∩ C nie może być zbiorem pustym: Y i Z mają krawędzie do zbioru C a więc gdyby S`∩ C było puste istniałaby ścieżka z Y-S do Z-S w G-S taka która zaczyna się w Y, wchodzi do C, przechodzi po C i na koniec wchodzi do Z. Dodatkowo |S`|<=|S|<w

Algorytm cd. 3:

3. Do drzewa dodajemy nowy węzeł s = (X ∪ S`), s staje się liściem t.

Po wykonaniu tego kroku niezmienniki nadal zachodzą:
1. Drzewo częściowej dekompozycji T pokrywa (U ∪ S`) wierzchołków. Spójna składowa C mogła rozpaść się na kilka mniejszych spójnych składowych (C` ⊂ C) grafu G-(U ∪ S`). Każda z tych spójnych składowych C` ma sąsiadów tylko w zbiorze (X ∪ S`), należy się upewnić, że mamy maksymalnie 3w sąsiadów. Przyjrzyjmy się pojedynczej spójnej składowej C`. Wszyscy sąsiedzi dla C` należą albo do ((X-Z) ∪ S`) albo do ((X-Y) ∪ S`) i każdy z tych zbiorów zawiera maksymalnie 3w wierzchołków. Jeżeli byłoby inaczej, czyli C` miałby sąsiada w Y-S oraz w Z-S a więc istniałaby ścieżka z Y-S do Z-S w G-S.

2. ponieważ wszyscy sąsiedzi S' są w X a dodatkowo |S`| < w i |X| =3w a więc |S` ∪ X| <=4w.

Drzewowa dekompozycja grafu – konstrukcja

Przedstawiony algorytm w każdym kroku zmniejsza liczbę wierzchołków grafu minimum o 1 a więc skończy się po maksymalnie n krokach.

Złożoność algorytmu:

- najbardziej kosztowną operacją jest sprawdzanie (w+1) spójności $O(f(w)m)$

- liczba kroków algorytmu $O(n)$

- sumaryczna złożoność $O(f(w)mn)$

funkcja f(w) jest funkcją wykładniczą rzędu 2^{6w}

Dziękuję.