Blum et al., Liniowa aproksymacja najkrótszego wspólnego nadsłowa, algorytm 3-przybliżony

Michał Zwonek

June 2020

Wprowadzenie

Mając na wejściu słowa $S = \{s_1, ..., s_m\}$, takie, że żadne z nich nie zawiera żadnego innego w całości, możemy poszukać wspólnego nadsłowo s dla danego zbioru S.

Najprostszym przykładem może być słowo $s = s_1 s_2 ... s_m$. Jednakże, tak uzyskane nadsłowo będzie niepotrzebnie długie.

Jako, że problem ten ma duże znaczenie praktyczne (w sekwencjonowaniu DNA) naturalnym jest próba znalezienia najkrótszego wspólnego nadsłowa.

Oczywistym podejściem jest następujący algorytm zachłanny. Po kolei łącz słowa mające największą część wspólną nachodzącą na siebie określaną jako *overlap*. Rób tak, aż do uzyskania jednego słowa. Postawiona jest hipoteza, że takie podejście ma współczynnik aproksymacji równy 2, ale przed opublikowaniem pracy przez Blum et al. nie było wiadome, czy taki algorytm jest w ogóle liniową aproksymacją. Okazuje się, że ten algorytm (określać będziemy go poprzez **GREEDY**) ma współczynnik aproksymacji co najwyżej 4, dowodu tego faktu nie będziemy przybliżać. Sam algorytm jest kluczowy, bo jest punktem startowym dla kolejnych dwóch algorytmów na których w tej pracy się skupimy.

Ponadto Blum et al. zaprezentowali alternatywny algorytm, nazywany **TGREEDY**, dla którego udowodnili, że jest on 3-aproksymacyjny. Sam problem znalezienia najkrótszego wspólnego nadsłowa jest NP trudny.

W pierwszej sekcji przedstawimy podstawowe definicje i notacje, a w kolejnych algorytm **MGREEDY** będący pewną modyfikacją algorytmu **GREEDY**, taką dla którego analiza aproksymacji jest prostsza. W następynej sekcji przedstawimy algorytm **TGREEDY** wykorzystujący algorytm **MGREEDY** by uzyskać pożądany współczynnik aproksymacji. W ostatniej sekcji postaramy się udzielić odpowiedzi na pytanie, który z tych dwóch algorytmów jest lepszy?

Podstawowe definicje

Mając dwa niekoniecznie różne słowa $s = uv, t = vw$, takie, że u, w są niepuste, a v maksymalnej długości, mianem *overlap*'u słów s, t określamy v. Długośc overlap'u słów s, t oznaczamy jako $ov(s, t) = |v|$. Zauważmy, że dla $s = t$ overlap jest maksymalnym właściwym prefikso-sufiksem.

Ponadto u nazwiemy *prefiksem słowa s względem słowa t* i oznaczymy poprzez $pref(s, t) = u$. *Odległością między s, a t* nazwiemy $d(s, t) = |pref(s, t)|$.

Wtedy słowo postaci $uvw = pref(s, t)t$ jest najkrótszym nadsłowem s i t o długości $d(s, t) + |t| = |s| + |t| - ov(s, t)$. Takie słowo będziemy też określali jako *merge słów s i t*. Dla słów $s_i, s_j \in S$ będziemy używali $pref(i, j), d(i, j), ov(i, j)$ zamiast $pref(s_i, s_j), d(s_i, s_j), ov(s_i, s_j)$.

Mając słowa $s_{i_1}, s_{i_2}, ..., s_{i_r}$ zdefiniujemy $s = \langle s_{i_1}, ..., s_{i_r} \rangle$ jako $s = pref(i_1, i_2)...pref(i_{r-1}, i_r)s_{i_r}$, czyli najkrótsze słowo, w którym słowa $s_{i_1}, ..., s_{i_r}$ pojawiają się dokładnie w tej kolejności. Określmy s_{i_1} poprzez $first(s)$ oraz s_{i_r} poprzez $last(s)$.

W trakcie algorytmu **GREEDY**, a także **MGREEDY** następujący niezmiennik jest zachowany.

Lemat 1. W dowolnym momencie algorytmu **GREEDY** w aktualnie przetwarzanym zbiorze słów dla każdych dwóch różnych słów s, t z tego zbioru jest, że $first(s)$ ani $last(s)$ nie jest podsłowem słowa t.

Proof. Udowodnimy lemat dla $first(s)$, dowód dla $last(s)$ jest zupełnie analogiczny.

Początkowo $first(s) = last(s) = s$ dla każdego słowa s w aktualnym zbiorze. Załóżmy, że niezmiennik jest w pewnym momencie naruszony poprzez złączenie dwóch słów t_1, t_2 w słowo $t = \langle t_1, t_2 \rangle$, czyli, że t ma $first(s)$ jako podsłowo. Wtedy wiemy, że dla $t_1 = ab, t_2 = bc$ (b jest maksymalne), $t = ufirst(s)v$ oraz $t = abc$. Skoro $first(s)$ nie może być podsłowem t_1 ani t_2 (założenie o zbiorze S), to wiemy, że $first(s)$ musi zawierać b jako podsłowo i nachodzić przynajmniej jedną literą na a i c. Z tego wynika, że $|u| < |a| = d(t_1, t_2), |v| < |c|$ oraz $|first(s)| > ov(t_1, t_2) = |b|$. Teraz możemy zauważyć, że $ov(t_1, s) \geq ov(t_1, first(s)) \geq |b| + 1 > ov(t_1, t_2) = |b|$, co stoi w sprzeczności z faktem, że $ov(t_1, t_2)$ miał być największą wartość. $\square$

Lemat ten mówi nam, że algorytm wybierając dwa słowa s, t o największym $ov(s, t)$ wybiera de facto słowa o największym $ov(first(s), last(t))$. W wyniku takiego łączenia powstaje słowo $\langle first(s), ..., last(s), first(t), ..., last(t) \rangle$. Możemy, więc powiedzieć, że **GREEDY** porządkuje słowa w ten sposób (dla każdej pary słów porównując ich overlap'y), a następnie znajduje najkrótsze wspólne nadsłowo zawierające te słowa w tej kolejności.

Wprowadzimy teraz pojęcie grafu odległości, który opisuje nam naszą instancję problemu. Mianowicie, $G_S = (V, E)$, gdzie wierzchołkom przypisane są poszczególne słowa z S, a krawędziom będą przypisane słowa $pref(i, j)$ o wadze $|pref(i, j)| = d(i, j)$.

Możemy na takim grafie łatwo stwierdzić, że jest zachowana nierówność: $CYC(G_S) \leq TSP(G_S) \leq OPT(S) - ov(last(s), first(s)) \leq OPT(S)$.

Dzięki temu można pokazać, że znajdując minmialne pokrycie cyklowe ($CYC(G_S)$) możemy skonstruować wspólne nadsłowo o długości co najwyżej $4 \cdot OPT(S)$ poprzez konkatenacje słów wynikających z poszczególnych cyklów w pokryciu cyklowym. Tak opisany algorytm nazywamy **CONCATCYCLES**.

Algorytm *MGREEDY*

Możemy teraz opisać algorytm **MGREEDY**.

1. Niech S będzie zbiorem słów początkowych, a T zbiorem pustym.
2. Dopóki S jest niepusty, wybierz s, t z S maksymalizujące $ov(s, t)$.
   Dla $s \neq t$, niech $S := (S \setminus \{s, t\}) \cup \{\langle s, t\rangle\}, T := T$.
   Dla $s = t$, niech $S := S \setminus \{s\}$, a $T := T \cup \{s\}$.
3. Jako wspólne nadsłowo zwróc konkatenację słów ze zbioru T.

Rozważmy teraz graf G'_S, analogiczny jak G_S, ale dla każdej krawędzi mamy jako wagę $ov(i, j)$ zamiast $d(i, j)$. Zauważmy, że dla dowonlego pokrycia cyklowego grafu G_S oraz tego samego pokrycia cyklowego grafu G'_S mamy, że suma wag z pokrycia cyklowego z tych dwóch grafów równa się sumie długości słów w S. Jest tak ponieważ $\forall s, t : ov(s, t) + d(s, t) = |s|$. Czyli maksymalne pokrycie cyklowe grafu G'_S, będzie tożsame z minimalnym pokryciem cyklowym grafu G_S. Okazuje się, że **MGREEDY** łącząc dwa słowa $s \neq t \in S$ w jedno, de facto wybiera krawędź w grafie G'_S między $s_\alpha := last(s)$, a $s_\beta := first(t)$ takie, że $ov(\alpha, \beta)$ jest maksymalne wśród wszystkich wolnych $ov(i, j)$.

Wystarczy teraz tylko pokazać, że taka zachłanna strategia dla tego rodzaju grafu konstruuje pokrycie cyklowe o maksymalnej wadze, co będzie oznaczało, że **MGREEDY** jest niegorszy niż **CONCATCYCLES**, czyli aproksymuje ze współczynnikiem 4. To też pokazali w swojej oryginalnej pracy Blum et al.

Algorytm *TGREEDY*

Algorytm **TGREEDY** najpierw wykonuje kroki od 1 do 2 z algorytmu **MGREEDY**, a następnie na uzyskanym zbiorze T, który odpowiada maksymalnemu pokryciu cyklowemu grafu G'_S, wykonuje algorytm **GREEDY**. Aby pokazać, że **TGREEDY** ma pożądany współczynnik aproksymacj potrzebujemy wprowadzić parę dodatkowych definicji i lematów.

Definicja 1. Niech $\equiv$ będzie relacją równoważności dla słów, taką, że $s \equiv t \Leftrightarrow \exists u, v : s = uv, t = vu$. Wtedy, jeżeli c to cykl na wierzchołkach (słowach) $i_0, ..., i_r$ to $strings(c)$ jest klasą równoważności $[pref(i_0, i_1)pref(i_1, i_2)...pref(i_{r-1}, i_0)]_\equiv$.
Ponadto, $strings(c, i_k)$ zdefiniujemy jako $pref(i_k, i_{k+1})...pref(i_{k-1}, i_k)$, gdzie indeksowanie jest modulo r. Oczywiście, $strings(c, i_k) \in strings(c)$.

Definicja 2. Niech $w(c)$, będzie wagą cyklu, czyli $w(c) = |s|, s \in strings(c)$. Dla wygody będziemy pisać, że $s \in c$ zamiast $s \in strings(c)$.

Lemat 2. Niech c_1, c_2 będą dwoma cyklami w minimalnym pokryciu cyklowym C, z $s_1 \in c_1$ oraz $s_2 \in c_2$ Wtedy, $ov(s_1, s_2) \leq w(c_1) + w(c_2)$.

Lemat 3. Nadsłowo $\langle s_{i_0}, ..., s_{i_{r-1}} \rangle$ jest podsłowem $strings(c, i_0)s_{i_0}$.

Twierdzenie. TGREEDY dla zbioru $S = \{s_1, ..., s_m\}$ zwraca nadsłowo s nad S o długości co najwyżej $3 \cdot OPT(S)$.

Proof. Niech $n := OPT(S)$, pokażemy, że $|s| \leq 3n$. Niech T będzie zbiorem samo-nachodzących się słów uzyskanych przez **MGREEDY** na S, a C będzie minimalnym pokryciem cyklowym skonstruowanym przez **MGREEDY**. Dla każdego $x \in T$, niech c_x oznacza cykl w C odpowiadający słowie x oraz niech $w_x = w(c_x)$ będzie jego wagą. Dla każdego zbioru słów R niech $||R|| = \sum_{x \in R} |x|$ będzie sumaryczną długością wszystkich słów w zbiorze R. Niech $w = \sum_{x \in T} w_x$. Skoro, $CYC(G_S) \leq TSP(G_S) \leq OPT(S)$, wiemy że $w \leq n$.

Dzięki lematowi 2, wiemy, że kompresja uzyskana w najkrótszym nadsłowie nad T jest mniejsza niż $2w$, i.e $||T|| - n_T \leq 2w$, gdzie $n_T := OPT(T)$. Ponadto, znanym rezultatem jest (Tarhio i Ukkonen, 1988), że algorytm zachłanny osiąga kompresję ze współczynnikiem 2. Łącząc te dwa wyniki dostajemy, że: $||T|| - |s| \geq (||T|| - n_T)/2 = ||T|| - n_T - (||T|| - n_T)/2 \geq ||T|| - n_T - w$.

Z tego wynika, że $|s| \leq n_T + w$.

Dla każdego $x \in T$, niech s_{i_x} będzie słowem z cyklu c_x, który jest prefiksem x. Niech $S' = \{s_{i_x} | x \in T\}, n' = OPT(S'), S'' = \{strings(c_x, i_x)s_{i_x} | x \in T\}, n'' = OPT(S'')$.

Z lematu 3 wynika, że nadsłowo S'' jest także nadsłowem T, więc $n_T \leq n''$. Dla dowolnej permutacji π prawdą jest, że $|S''_\pi| \leq |S'_\pi| + \sum_{x \in T} w_x$, więc $n'' \leq n' + w$, gdzie S'_π, S''_π jest nadsłowem uzyskanym poprzez połączenie słów z S', S'' w kolejności narzuconej przez π.

Wiemy, że $S' \subset S''$, a więc $n' \leq n$. Możemy teraz połączyć te wyniki i dostać, że

$$n_T \leq n'' \leq n' + w \leq n + w$$

Łącząc to z faktem, że $|s| \leq n_T + w$ dostajemy, że $|s| \leq n + 2w \leq 3n$. □

Uwagi końcowe

Blum et al. pokazali dodatkowo, że **GREEDY** osiąga współczynnik aproksymacji równy 4, ale hipoteza czy ten współczynnik można obniżyć do 2 jest wciąż otwarta. Możnaby się pokusić wysunąć hipotezę, że **TGREEDY** jest lepszy niż **GREEDY**. Tak nie jest. Są przykłady zbioru S dla których zarówno jeden jak i drugi wypada lepiej.

Dla zbioru $S = \{c(ab)^k, (ab)^{k+1}, (ba)^k c\}$ **TGREEDY** wygeneruje słowo $c(ab)^k ac(ab)^{k+1} a$ o długości $n = 4k+6$, podczas gdy **GREEDY** zwróci słowo optymalne $c(ab)^{k+1} ac$ o długości $n = 2k + 5$.

Z kolei dla zbioru $S = \{cab^k, ab^k ab^k a, b^k dab^{k-1}\}$ **TGREEDY** wygeneruje słowo $cab^k dab^k ab^k a$ o długości $3k+6$, podczas gdy **GREEDY** wygeneruje słowo $cab^k ab^k ab^k dab^{k-1}$ długości $4k+5$.