Algorytm 2.89-aproksymacyjny
dla problemu najkrótszego wspólnego nadsłowa
autorstwa Shang-Hua Teng oraz Frances F. Yao [1]

1 Notacja i definicje

Dla danego na wejściu zbioru słów S chcemy znaleźć najkrótsze słowo α będące nadsłowem tego zbioru, t.j. $\forall_{s \in S}\, s$ jest podsłowem α. Standardowo, zakładamy że dla żadnej pary słów nie zachodzi sytuacja, w której jedno jest podsłowem drugiego. Wprowadzamy następujące definicje i oznaczenia:

- $|s|$ jest długością słowa s,
- $|S| = \sum_{s \in S} |s|$,
- $opt(S)$ jest długością wyniku optymalnego - najkrótszego nadsłowa słów ze zbioru S.

Dla każdej pary słów $s = uv$, $t = vw$:

- $ov(s,t) = |v|$ jest długością "nakładania się" na siebie słów s i t,
- $pref(s,t) = u$.

Dla takich słów s i t najkrótszym ich nadsłowem, w którym s występuje przed t jest $uvw = pref(s,t)t$. Słowo takie nazywamy złączeniem słów s i t oraz oznaczamy je przez $\langle s,t \rangle$. Łatwo zauważyć, że najkrótsze nadsłowo α jest równe $\langle s_{\pi(1)}, s_{\pi(2)}, \ldots, s_{\pi(n)} \rangle$ dla pewnej permutacji π zbioru $\{1, 2, \ldots, n\}$.

Problem znajdowania najkrótszego nadsłowa może zostać przeformułowany na problem grafowy. Dla zbioru słów S możemy stworzyć ważony graf skierowany $G = (S, E)$, w którym wagi krawędzi są równe $d(s,t) = |pref(s,t)|$. Dla zbioru krawędzi zdefiniujmy $d(E) = \sum_{(s,t) \in E} d(s,t)$ i analogicznie $ov(E) = \sum_{(s,t) \in E} ov(s,t)$. Prosta obserwacja pokazuje, że suma długości słów w cyklu E jest równa $d(E) + ov(E)$.

2 Optymalne pokrycie cyklowe

Pokryciem cyklowym grafu G jest zbiór cykli taki, że każdy wierzchołek G należy do któregoś z nich. Optymalne pokrycie cyklowe jest pokryciem cyklowym z (w tym przypadku) najmniejszą możliwą sumą wag krawędzi. Można je znaleźć w czasie $O(n^3)$ algorytmem węgierskim [1].

Niech $C = \{c_1, c_2, \ldots, c_k\}$ będzie pokryciem cyklowym. By udowodnić współczynnik aproksymacji należy wprowadzić najpierw kilka lematów.

[1] `https://en.wikipedia.org/wiki/Hungarian_algorithm`

Lemat 2.1. *$d(C) \leqslant opt(S)$ oraz $ov(C) \geqslant |S| - opt(S)$.*

Lemat 2.2. *Niech $ov_m = \sum ov(C)$ dla cykli składających się z m wierzchołków. Niech C' będzie podzbiorem C powstałym przez usunięcie krawędzi z najmniejszym $ov(s,t)$ z każdego cyklu. Wtedy $ov(C') \geqslant ov_2/2 + ov_3/3 + ov_4/4 + \dots$.*

Warto zauważyć, że nadsłowo powstałe przez złączenie ze sobą wszystkich ścieżek z C' uzyskuje $ov(C') \geqslant ov(C)/2$, a więc jest co najmniej tak dobre jak nadsłowo stworzone algorytmem zachłannym.

Lemat 2.3 ([2])**.** *Niech $c_1, c_2 \in C$ oraz $s_1 \in c_1, s_2 \in c_2$. Wtedy $ov(s_1, s_2) \leqslant d(c_1) + d(c_2)$.*

Dla słowa s_i znajdującego się w cyklu $c \in C$ wprowadźmy oznaczenie $\langle s_i, c\rangle = \langle s_i, s_{i+1}, \dots, s_m, s_1, \dots, s_i\rangle$. Stwórzmy zbiór R wybierając dowolne słowo z każdego cyklu z C. Niech $\alpha = \langle r_1, \dots, r_k\rangle$ będzie nadsłowem słów z R. Słowo $\overline{\alpha} = \langle\langle r_1, c_1\rangle, \dots, \langle r_k, c_k\rangle\rangle$ będziemy nazywać rozszerzeniem słowa α - jest ono nadsłowem wszystkich słów z S.

Lemat 2.4. *$|\overline{\alpha}| = |\alpha| + d(C)$.*

2.1 Kanoniczne pokrycie cyklowe

Dla cyklu $c = (s_1, s_2, \dots, s_m) \in C$ definiujemy $period(c) = pref(s_1, s_2)pref(s_2, s_3)\dots pref(s_m, s_1)$. Mówimy, że słowo s pasuje do cyklu c jeśli s jest podsłowem $period(c)^k$ dla pewnego k. Oczywiście, jeśli s należy do c, to s pasuje do c.

s może jednak pasować do innych cykli. Niech s należy do c ale pasuje również do c'. Możemy wtedy “przenieść” s do c' nie zmieniając sumy wag wszystkich krawędzi w obu cyklach. Kanoniczne pokrycie cyklowe to takie pokrycie cyklowe, w którym każdy wierzchołek s jest przyporządkowany cyklowi o najmniejszej wadze wśród wszystkich, do których s pasuje. W pokryciu takim zachodzi następująca własność:

Lemat 2.5. *Niech $c_1, c_2 \in C$, gdzie C jest kanoniczne oraz $s_1 \in c_1, s_2 \in c_2$. Wtedy $ov(s_1, s_2) + ov(s_2, s_1) < max(|s_1|, |s_2|) + min(d(c_1), d(c_2))$.*

3 Greedy-Insert

Niech C_2 będzie zbiorem cykli 2-elementowych. Nadsłowo wszystkich słów należących do C_2 tworzymy używając algorytmu Greedy-Insert:

1. Podziel C_2 na zbiory F i G. F zawiera krótsze słowa z każdego cyklu, G dłuższe.

2. $(f_1, f_2, \dots, f_n)$ - słowa z F w dowolnej kolejności,
   $(g_1, g_2, \dots, g_n)$ - słowa z G w kolejności w jakiej występują w nadsłowie η otrzymanym standardowym algorytmem zachłannym.

3. $q_O = \langle f_1, g_1, g_2, f_2, f_3, g_3, g_4, f_4, \dots\rangle$
   $q_E = \langle g_1, f_1, f_2, g_2, g_3, f_3, f_4, g_4, \dots\rangle$.

4. Zwróć słowo q - krótsze ze słów q_O, q_E.

Zbiór sąsiadujących ze sobą słów w q_O i q_E jest nadzbiorem sąsiadujących ze sobą słów w C_2 i η. Wybierając lepsze ze słów q_O, q_E otrzymujemy następujący lemat:

Lemat 3.1. $ov(q) \geqslant ov(C_2)/2 + ov(\eta)/2$.

Korzystając z własności algorytmu zachłannego ($ov(\eta) \geqslant (|G| - opt(G))/2$):

Lemat 3.2. $ov(q) \geqslant ov(C_2)/2 + (|G| - opt(G))/4$.

4 Opis algorytmu

Algorytm w całości prezentuje się następująco:

Wejście: $S = \{s_1, s_2, \ldots, s_n\}$.

1. Znajdź optymalne pokrycie cyklowe C w S i przekształć je w pokrycie kanoniczne.
2. Z każdego cyklu wybierz dowolne słowo tworząc zbiór R.
3. Znajdź optymalne pokrycie cyklowe CC dla zbioru R.
4. Uruchom Greedy-Insert na wszystkich cyklach długości 2, otrzymując słowo q.
5. We wszystkich pozostałych cyklach usuń krawędź z najmniejszym nakładaniem się słów i połącz wszystkie ścieżki w słowo α.
6. Zwróć $\overline{\alpha}$ - rozszerzenie słowa α.

5 Analiza algorytmu

Wprowadźmy oznaczenie d_2, d_3, d_4 na sumę wag krawędzi dla wszystkich cykli długości odpowiednio $2, 3$ oraz 4 lub większych. Analogicznie ov_2, ov_3, ov_4 na $\sum ov(C)$ dla cykli długości $2, 3$ oraz 4 lub większych.

Z lematu 2.5, sumując po wszystkich cyklach:

Lemat 5.1. $ov_2 \leqslant |G| + d_2/2$.

Twierdzenie 5.1. $|\alpha| \leqslant (1 + 8/9)opt(S)$.

Dowód. Z lematu 2.2 i 3.2:

$$ov(\alpha) \geqslant ov_2/2 + (|G| - opt(G))/4 + 2ov_3/3 + 3ov_4/4$$

Z lematu 2.3 $ov_i \leqslant 2d_i$. Z lematu 2.1 wiemy też, że $|R| \leqslant opt(R) + ov(CC) \leqslant opt(S) + ov(CC)$.

$$\begin{aligned}
|\alpha| &= |R| - ov(\alpha) \\
&\leqslant opt(S) + ov(CC) - ov_2/2 - 2ov_3/3 - 3ov_4/4 \\
&= opt(S) + ov_2/2 + ov_3/3 + ov_4/4 \\
&\leqslant opt(S) + d_2 + (2/3)d_3 + d_4/2
\end{aligned}$$

Możemy też inaczej ograniczyć od góry $|\alpha|$. Korzystamy m.in. z lematu 5.1.

$$\begin{aligned}
|\alpha| &= |R| - ov(\alpha) \\
&\leqslant |R| - ov_2/2 - (|G| - opt(G))/4 - 2ov_3/3 - 3ov_4/4 \\
&\leqslant |R| - ov_2/2 - (ov_2 - d_2/2 - opt(S))/4 - 2ov_3/3 - 3ov_4/4 \\
&= |R| - 3ov_2/4 - 2ov_3/3 - 3ov_4/4 + d_2/8 + opt(S)/4 \\
&= 5opt(S)/4 + ov_2/4 + ov_3/3 + ov_4/4 + d_2/8 \\
&= 5opt(S)/4 + ov(CC)/4 + ov_3/12 + d_2/8 \\
&= (2 - 1/4)opt(S) + d_3/6 + d_2/8
\end{aligned}$$

Analizę algorytmu kończymy rozważając dwa przypadki:

1. $d_2 \leqslant (2/3)d(C)$, wtedy z pierwszej nierówności:

$$\begin{aligned}
|\alpha| &\leqslant opt(S) + (2/3)d(C) + (2/3)(1/3)d(C) \\
&\leqslant (1 + 8/9)opt(S)
\end{aligned}$$

2. $d_2 > (2/3)d(C)$, wtedy $d_3 \leqslant (1/3)d(C)$ i z drugiej nierówności:

$$\begin{aligned}
|\alpha| &\leqslant (2 - 1/4)opt(S) + d_3/6 + d_2/8 \\
&= (2 - 1/4)opt(S) + (d_2 + d_3)/8 + d_3/24 \\
&\leqslant (2 - 1/4)opt(S) + d(C)/8 + d(C)/72 \\
&\leqslant (1 + 8/9)opt(S)
\end{aligned}$$

□

Z lematu 2.4. $|\overline{\alpha}| = |\alpha| + d(C) \leqslant (2 + 8/9)opt(S)$, co kończy dowód współczynnika aproksymacji.

Literatura

[1] S.-H. Teng, F.F. Yao. *Approximating shortest superstrings.* SIAM J. Comput., 1997, Volume 26(2), pp. 410–417.

[2] A. Blum, T. Jiang, M. Li, J. Tromp, M. Yannakakis. *Linear approximation of shortest superstrings.* Proc. 23rd ACM Symposium on the Theory of Computing, ACM, New York, 1991, pp. 328–336.