Query-by-Sketch: Scaling Shortest Path Graph Queries on Very Large Networks

Paweł Polerowicz

Styczeń 2024

1 Wiadomości wstępne

1.1 Informacje o artykule

- Tytuł: Query-by-Sketch: Scaling Shortest Path Graph Queries on Very Large Networks
- Autorzy: Ye Wang, Qing Wang, Henning Koehler, Yu Lin
- Data publikacji: 2021
- miejsce publikacji: Proceedings of the 2021 International Conference on Management of Data

1.2 Słowniczek pojęć

Oznaczenie	Znaczenie
$G(V, E)$	graf (tu nieskierowany i spójny)
P_{uv}	zbiór najkrótszych ścieżek między wierzchołkami u i v
$d_G(u, v)$	długość najkrótszej ścieżki między u i v
$R \subset V$	zbiór punktów orientacyjnych (ang. *Landmarks*)
$L(v) = \{(r_1, \delta_{vr_1}), \cdots, (r_n, \delta_{vr_n})\}$	zbiór etykiet wierzchołka v
δ_{vr_i}	$= d_G(v, r_i)$
$L = \{L(v)\}_{v \in V}$	etykietowanie nad G
$size(L) = \sum_{v \in V} \lvert L(v) \rvert$	rozmiar etykietowania

1.3 Plan prezentacji

W ramach niniejszej prezentacji zreferujemy artykuł *Query-by-Sketch: Scaling Shortest Path Graph Queries on Very Large Networks*, przedstawiający metodę QbS. Składa się ona z trzech algorytmów. Pierwszy z nich służy jako przetwarzanie wstępne i są wykonywany raz i wyznacza etykietowanie L dla grafu G. Drugi tworzy szkic danych dla pary wierzchołków u i v na podstawie etykietowania L. Ostatni wyznacza dokładną odpowiedź, wykorzystując przeszukiwanie kierowane szkicem. Złożoności algorytmów to kolejno $O(|R||E|)$, $O(|R|^4)$ (z możliwością ograniczenia do $O(|R|^2)$) oraz $O(|E| + |R||V|)$, gdzie R jest zbiorem punktów orientacyjnych.

1.4 Motywacja

Podczas poprzednich prezentacji omawialiśmy struktury wykorzystujące szkice danych do efektywnego pamięciowo przechowywania informacji o strumieniowanym grafie. W szczególności rozważaliśmy takie konstrukcje, które pozwalały na uzyskiwanie szybkich odpowiedzi na zapytania o istnienie i wagę krawędzi między dwoma wierzchołkami, a także łączną wagę krawędzi wchodzących i wychodzących z danego wierzchołka. Niemniej jednak często zależy nam na wykonywaniu bardziej skomplikowanych operacji. Jedną z typowych może być wyznaczanie najkrótszych ścieżek między wierzchołkami. Rozwiązania tego problemu mogą znaleźć swoje zastosowanie np. w nawigacji GPS lub w analizie sieci społecznościowych. W obu przypadkach mamy do czynienia z grafami o nierzadko ogromnych rozmiarach. Sprawia to, że klasyczne i dobrze przebadane algorytmy mogą okazać się nieskuteczne, ze względu na konieczność przeglądu zatrważającej liczby krawędzi lub wykorzystywania dodatkowej pamięci o niepraktycznym rozmiarze. Dodatkowo niejednokrotnie możemy być zainteresowani wyborem nie jednej ścieżki, lecz pewnego zbioru najkrótszych lub prawie najkrótszych ścieżek. Śledząc wysiłki autorów artykułu, spróbujemy znaleźć metodę wyznaczania grafu najkrótszych ścieżek w sposób wydajny czasowo i przy użyciu rozsądnej ilości pamięci.

2 Problem

2.1 Sformułowanie problemu

2-hop distance cover (dwu-przeskokowe pokrycie odległości :))

Etykietowanie L grafu G stanowi 2-hop distance cover, jeśli zachodzi

$$\forall_{u,v\in V} d_G(u,v) = min\{\delta_{ur} + \delta_{vr} : (r, \delta_{ur}) \in L(u), \delta_{vr} \in L(v)\}$$

Innymi słowy, wymagamy, aby dla każdej pary wierzchołków, ich etykiety zawierały co najmniej jeden wspólny punkt orientacyjny leżący na jednej z najkrótszych ścieżek je łączących.

SPG (graf najkrótszych ścieżek)

Dla dowolnych dwóch wierzchołków u i v, graf najkrótszych ścieżek (SPG) między u i v to podgraf G_{uv} grafu, gdzie:

1. $V(G_{uv}) = \bigcup_{p\in P_{uv}} V(p)$
2. $E(G_{uv}) = \bigcup_{p\in P_{uv}} E(p)$

Graf ten nie jest więc po prostu podgrafem indukowanym przez $\bigcup_{p\in P_{uv}} V(p)$, gdzie P_{uv} to zbiór najkrótszych ścieżek między wierzchołkami u i v. Każda jego krawędź musi być bowiem częścią najkrótszej ścieżki między u i v.

Problem SPG

Niech $G = (V, E)$ oraz $u, v \in V$. Problem SPG polega na znalezieniu odpowiedzi na zapytanie $SPG(u, v)$, czyli grafu najkrótszych ścieżek G_{uv} dla G. W niniejszej prezentacji będziemy dla uproszczenia zakładali, że graf jest nieskierowany, spójny i nie jest ważony. W ogólności jednak te ograniczenia są możliwe do zrelaksowania.

Algorithm 2: Constructing a labelling scheme $\mathcal{L}$

Input: $G = (V, E)$; a set of landmarks $R \subseteq V$

Output: A labelling scheme $\mathcal{L} = (M, L)$ with $M = (R, E_R, \sigma)$.

```
E_R ← ∅; L(v) ← ∅ for all v ∈ V
for all r_i ∈ R do
    Q_L ← ∅; Q_N ← ∅;
    Q_L.push(r_i);
    depth[r_i] ← 0; depth[v] ← ∞ for all v ∈ V\{r_i};
    n = 0;
    while Q_L and Q_N are not empty do
        for all u ∈ Q_L at depth n do
            for all unvisited neighbors v of u do
                depth[v] ← n + 1;
                if v is a landmark then
                    Q_N.push(v);
                    E_R ← E_R ∪ {(r_i, v)};
                    σ(r_i, v) ← depth[v];
                else
                    Q_L.push(v);
                    L(v) ← L(v) ∪ {(r_i, depth[v])};
        for all u ∈ Q_N at depth n do
            for all unvisited neighbors v of u do
                depth[v] ← n + 1;
                Q_N.push(v);
        n ← n + 1;
```

Rysunek 1: Pseudokod procedury etykietowania

2.2 Główne idee rozwiązań

3 Przegląd poprzednich rozwiązań

3.1 Klasyczne algorytmy

3.2 Algorytmy dokładne dla dużych grafów

3.3 Algorytmy aproksymacyjne

4 QbS

4.1 Idea

4.2 Przykład

4.3 Etykietowanie

4.4 Szkicowanie grafu najkrótszych ścieżek

4.5 Przeszukiwanie kierowane

Algorithm 3: Computing a sketch S_{uv}

Input: $\mathcal{L} = (M, L)$, two vertices u and v.

Output: A sketch $S_{uv} = (V_S, E_S, \sigma_S)$

$V_S \leftarrow \emptyset, E_S \leftarrow \emptyset$;

for *all* $\{r, r'\} \subseteq R$ **do**

    $\pi_{rr'} \leftarrow +\infty$;

    if $(r, \delta_{ur}) \in L(u)$ *and* $(r', \delta_{vr'}) \in L(v)$ **then**

        $\pi_{rr'} \leftarrow \delta_{ur} + d_M(r, r') + \delta_{vr'}$;

$d^{\top}_{uv} \leftarrow \min\{\pi_{rr'} | \{r, r'\} \subseteq R\}$;

for *all* $\{r, r'\} \subseteq R$ *and* $\pi_{rr'} = d^{\top}_{uv}$ **do**

    $E_S \leftarrow E_S \cup \{(u, r), (v, r')\}$;

    $\sigma_S(u, r) \leftarrow \delta_{ur}, \sigma_S(v, r') \leftarrow \delta_{vr'}$;

    for *all* (r_i, r_j) *in the shortest path graph of* (r, r') *in* M **do**

        $E_S \leftarrow E_S \cup \{(r_i, r_j)\}$;

        $\sigma_S(r_i, r_j) \leftarrow \sigma(r_i, r_j)$;

    $V_S \leftarrow V(E_S)$;

Rysunek 2: Pseudokod procedury szkicowania

Algorithm 4: Searching on $G[V\backslash R]$

Input: $G^{-} = G[V\backslash R], S_{uv}, \mathcal{L} = (M, L)$

Output: A shortest path graph G_{uv}

$d_{uv}^{\top}, d_{u}^{*}, d_{v}^{*} \leftarrow get\_bound(S_{uv})$;

$P_u \leftarrow \emptyset, P_v \leftarrow \emptyset, d_u \leftarrow 0, d_v \leftarrow 0$;

Enqueue u to Q_u and v to Q_v;

$depth_u[w] \leftarrow \infty, depth_v[w] \leftarrow \infty$ for all $w \in V\backslash R$;

$depth_u[u] \leftarrow 0, depth_v[v] \leftarrow 0$;

while $d_u + d_v < d_{uv}^{\top}$ **do**

    $t \leftarrow pick\_search(P_u, P_v, d_u^{*}, d_v^{*}, d_u, d_v)$;

    if $t = u$ **then**

        $Q_u \leftarrow forward\_search(Q_u)$;

    if $t = v$ **then**

        $Q_v \leftarrow backward\_search(Q_v)$;

    $P_t \leftarrow P_t \cup Q_t$; $d_t \leftarrow d_t + 1$;

    $depth_t[w] \leftarrow d_t$ **for** $w \in Q_t$;

    if $P_u \cap P_v$ *is not empty* **then**

        break;

if $P_u \cap P_v \neq \emptyset$ **then**

    $G_{uv}^{-} \leftarrow reverse\_search(P_u \cap P_v, G^{-}, depth_u, depth_v)$;

if $d_u + d_v = d_{uv}^{\top}$ **then**

    $Z \leftarrow \emptyset$;

    for *all* $(r, t) \in E_S$ *with* $t \in \{u, v\}$ **do**

        $d_m \leftarrow \min\{\sigma_S(r, t) - 1, d_t\}$;

        for *all* w *with* $depth_t[w] = d_m, (r, \delta_{wr}) \in L(w), \delta_{wr} + d_m = \sigma_S(r, t)$ **do**

            $Z \leftarrow Z \cup \{(w, r)\}$;

    $G_{uv}^{\mathcal{L}} \leftarrow recover\_search(S_{uv}, \mathcal{L}, Z, G^{-}, depth_u, depth_v)$;

$G_{uv} \leftarrow G_{uv}^{-} \cup G_{uv}^{\mathcal{L}}$;

Rysunek 3: Pseudokod przeszukiwania kierowanego szkicem