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. Jest 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.

2.2 Główne idee rozwiązań

3 Przegląd poprzednich rozwiązań

3.1 Klasyczne algorytmy

Algorytm Dijkstry - $O(|E| + |V|log(|V|))$ dla grafów ważonych. BFS - $O(|E|)$ dla nieważonych. Nieefektywne na wielkich grafach.

3.2 Algorytmy dokładne dla dużych grafów

Np. PLL - etykietowanie 2-hop. IS-label - dla ważonych grafów, oparte o niezależne zbiory wierzchołków.

3.3 Algorytmy aproksymacyjne

Niektórzy naukowcy porzucili wymagania co do dokładności odpowiedzi na rzecz aproksymacji. Jendym z pomysłów było np. wstępne wyliczenie ścieżek z każdego wierzchołka do każdego punktu orientacyjnego, a następnie zwracanie połączonych ścieżek do tychże punktów.

4 QbS

4.1 Idea

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

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**
        $\pi_{rr'} \leftarrow \delta_{ur} + d_M(r, r') + \delta_{vr'}$;
$d_{uv}^{\top} \leftarrow \min\{\pi_{rr'} | \{r, r'\} \subseteq R\}$;
for *all* $\{r, r'\} \subseteq R$ *and* $\pi_{rr'} = d_{uv}^{\top}$ **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

4.2 Przykład

4.3 Etykietowanie

Niech $G = (v, E)$ - graf, $R \subset V$ - zbiór punktów orientacyjnych spełniający $|R| \ll |V|$. Schemat QbS zaczyna się od przetwarzania wstępnego, którego celem jest znalezienie kompaktowej reprezentacji najkrótszych ścieżek pomiędzy punktami orientacyjnymi, zwanej meta-grafem grafu G. Następnie, bazując na tymże meta-grafie, tworzymy etykietowanie, każdemu wierzchołkowi przypisując każdemu wierzchołkowi taką etykietę, aby móc efektywnie policzyć szkic dla zapytania $SPG(u, v)$ dla dowolnych u i v.

Meta-graf

Meta grafem nazywamy trójkę $M = (R, E_R, \sigma)$, gdzie $E_R \subset R \times R$ jest zbiorem krawędzi postaci (r, r') spełniających warunek, że przynajmniej jedna najkrótsza ścieżka między r i r' w oryginalnym nie przechodzi przez żaden inny punkt orientacyjny. $\sigma : E_R \to \mathbb{N}$ przypisuje każdej krawędzi w E_R wagę równą długości najkrótszej ścieżki. Przykład ukazano na rysunku 4.

System etykietowania (ang. *Labelling scheme*)

System etykietowania $\mathcal{L} = (M, L)$ składa się z meta-grafu M oraz etykietowania, które przypisuje każdemu wierzchołkowi $u \in V \setminus R$ etykietę $L(u)$ taką że:

$$L(u) = \{(r, \delta_{ur}) : r \in R, \delta_{ur} = d_G(u, r), (\exists p \in P_{ur})(V(p) \cap Rr)\}$$

Innymi słowy, para (r, δ_{ur}) jest częścią etykiety $L(u)$ tylko, jeśli istnieje co najmniej jedna najkrótsza ścieżka między u i r, która nie zawiera innych punktów orientacyjnych.

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

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

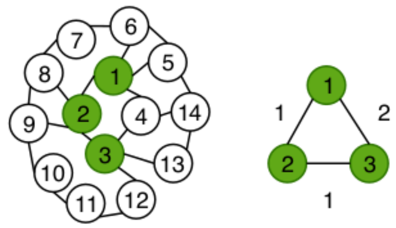
Output: A shortest path graph G_{uv}

$d_{uv}^{\top}, d_u^*, d_v^* \leftarrow get\_bound(S_{uv})$;
$P_u \leftarrow \emptyset, P_v \leftarrow \emptyset, d_u \leftarrow 0, d_v \leftarrow 0$;
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

Algorytm 1

Algorytm etykietowana ukazano na obrazku 1. Wykonujemy BFS dla każdego punktu orientacyjnego r. Wykorzystujemy dwie kolejki do przechowywania odwiedzonych wierzchołków, zależnie od tego, czy mają one zostać zaetykietowane z wykorzystaniem r, czy też nie. Wszystkie wierzchołki oprócz r są na początku niezaetykietowane. Dla każdego wierzchołka w kolejce zaetykietowanych z r na n-tym poziomie BFS rozpatrujemy nieodwiedzonych sąsiadów i oznaczamy ich jako odwiedzonych. Jeśli dany sąsiad jest punktem orientacyjnym to dodajemy krawędź do meta-grafu i ustawiamy głębokość jako dystans w funkcji σ. W przeciwnym przypadku dodajemy parę $(r, depth(v))$ do etykiety $L(V)$, a sam wierzchołek v do kolejki wierzchołków zaetykietowanych z r. Następnie rozpatrujemy kolejkę niezaetykietowanych na n-tym poziomie i wszystkich nieodwiedzonych sąsiadów dodajemy do kolejki niezaetykietowanych. Powtarzamy na kolejnych poziomach aż do przetworzenia wszystkich wierzchołków. Przykład ukazano na rysunku 4

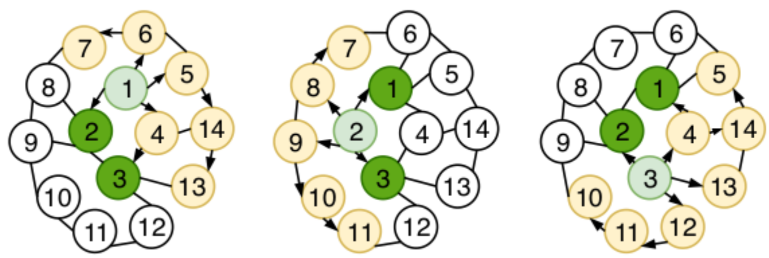


(a) (b)

Label	Labelling Entries
L(4)	(1,1)(3,1)
L(5)	(1,1)(3,3)
L(6)	(1,1)
L(7)	(1,2)(2,2)
L(8)	(2,1)
L(9)	(2,1)
L(10)	(2,2)(3,3)
L(11)	(2,3)(3,2)
L(12)	(3,1)
L(13)	(1,3)(3,1)
L(14)	(1,2)(3,2)

(c)

Rysunek 4: Pseudokod przeszukiwania kierowanego szkicem



Rysunek 5: Przykład etykietowania kolejno dla wierzchołków 1, 2 i 3

4.4 Szkicowanie grafu najkrótszych ścieżek

Aby móc efektywnie odpowiedzieć na zapytanie $SPG(u,v)$ niezbędne jest przygotowanie szkicu danych dla wierzchołków u i v. Będzie on oparty o schemat $\mathcal{L}$.

Szkic

Szkicem dla $SPG(u,v)$ na podstawie $\mathcal{L}$ nazywamy trójkę $S_{uv} = (V_S, E_S, \sigma_S)$, gdzie $V_S = \{u,v\} \cup R$ jest zbiorem wierzchołków, E_S zbiorem krawędzi, a $\sigma_S \to \mathbb{N}$ zachowuje $\sigma_S(u',v') = d_G(u',v')$. Trójka ta zachowuje warunek, że E_S zawiera jedynie krawędzie na ścieżkach między u i v o minimalnej długości, a więc:

$$d^T_{uv} = min_{(r,r')}\{\delta_{ru} + d_M(r,r') + \delta_{r'v} : (r,\delta_{ru}) \in L(u), (r',\delta_{r'v}) \in L(v)\}$$

Oczywiście, zachodzi własność $d^T_{uv} \geqslant d_G(u,v)$.

Algorytm 2

Algorytm ukazany na rysunku 3 tworzy omawiany szkic. Najpierw, zbiory wierzchołków i krawędzi w szkicu są puste. Zaczynamy od sprawdzenia wszystkich par punktów orientacyjnych, wyznaczając najkrótszą ścieżkę z u do v przechodzącą przez oba te punkty używając etykiet i meta-grafu. Najkrótsza otrzymana ścieżka to d^T_{uv}. Następnie, dla wszystkich wyznaczonych ścieżek o takiej długości, dodajemy ich krawędzie do zbioru krawędzi szkicu, łącznie z krawędziami łączącymi u i v z punktami obserwacyjnymi. Analogicznie, do zbioru dodajemy wszystkie wierzchołki na tych ścieżkach.

4.5 Przeszukiwanie kierowane

Ostatnim i zasadniczym krokiem w procesie odpowiedzi na zapytanie jest właściwe przeszukiwanie. Aby ograniczyć liczbę przetwarzanych krawędzi, będziemy korzystać ze szkicu S_{uv} utworzonego w poprzednim kroku. Dodatkowo przeszukiwanie możemy przeprowadzić na grafie $G[V \setminus R]$, a więc bez punktów orientacyjnych i krawędzi do nich prowadzących. Przy dobrym wyborze punktów orientacyjnych może to zauważalnie ograniczyć rozmiar przeszukiwanej przestrzeni. Oczywiście długość najkrótszej ścieżki w takim ograniczonym grafie może być większa niż w oryginalnym, ale liczbę kroków przeszukiwania możemy ograniczyć przez obliczoną wcześniej wartość d^T_{uv}. S_{uv} pozwala na wyznaczenie dwukierunkowej sekwencji poszukiwań. Konkretnie, dla $t \in \{u, v\}$ mamy

$$d^*_t = max_{(r,t)\in E_S} \sigma_S(r, t) - 1$$

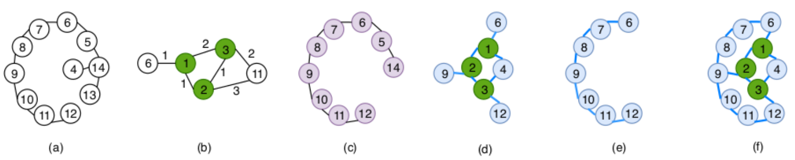
Wartość ta wyznacza liczbę kroków poszukiwań ze strony u i v. Przeszukiwanie ma trzy fazy:

1. Dwukierunkowe przeszukiwanie BFS - 'naprzód' dla wierzchołka u i 'wstecz' dla wierzchołka v. Kończy się ona po napotkaniu wspólnych wierzchołków lub przekroczeniu limitu kroków.

2. Przeszukiwanie odwrotne, ktore odwraca poprzednie kroki, aby skontruować najkrótsze ścieżki w G^-.

3. Wyszukiwanie odzyskujące (ang. *Recovery search*) konstruujące ścieżki przechodzące przez R.

Oczywiście dwa ostatnie kroki wykonujemy kondycjonalnie, zależnie od wyniku pierwszego. Przechowujemy kolejki P_u i P_v, przechowujące wierzchołki odwiedzone w przeszukiwaniu odpowiednio z u i v oraz głebokości poszukiwań d_u i d_v. Kolejki Q_u i Q_v zawierają wierzchołki odwiedzone na poziomach d_u i d_v. Na początku P_u i P_v są puste. W każdej iteracji pętli pierwszego kroku wybieramy u albo v bazując na głębokości poszukiwań i rozmiarze kolejek i powtarzamy do momentu znalezienia najkrótszych ścieżek lub osiągnięcia limitu. Jeśli znaleźliśmy w ten sposób ścieżkę, to wykonujemy przeszukiwanie wstecz. Na końcu sprawdzamy ścieżki zawierające punkty orientacyjne. Przykład zaprezentowano na rysunku 6.



Rysunek 6: Pseudokod procedury etykietowania

Algorytm 3

Odpowiedź G_{uv} może więc być wyznaczona w następujący sposób (ozn, $G^- = G[V \setminus R]$).

$$G_{uv} = \begin{cases} G^{\mathcal{L}}_{uv} & \text{jeśli } d_{G^-}(u, v) > d^T_{uv}, \\ G^{\mathcal{L}}_{uv} \cup G^-_{uv} & \text{jeśli } d_{G^-}(u, v) = d^T_{uv}, \\ G^-_{uv} & wp.p. \end{cases}$$

gdzie $G^{\mathcal{L}}_{uv}$ na najkrótsza ścieżka przechodząca przez co najmniej jeden punkt orientacyjny.

4.6 Poprawność (szkic)

1. Należy pokazać, że wyznaczone etykietowanie spełnia definicję systemu etykietowania, co sprowadza się do zauważania, że algorytm do etykiety każdego wierzchołka dodaje tylko i wyłącznie te punkty orientacyjne, do których najkrótsza ścieżka nie prowadzi przez inne punkty orientacyjne. Jest to oczywiste po przeanalizowaniu pseudokodu.

2. Trzeba udowodnić, iż szkic skonstruowany w drugim kroku QbS spełnia definicję szkicu dla $SPG(u, v)$. Jest to dość oczywiste, gdyż algorytm najpierw znajduje parę punktów minimalizującą $\{\delta_{ru} + d_M(r, r') + \delta_{r'v} : (r, \delta_{ru}) \in L(u), (r', \delta_{r'v}) \in L(v)\}$, a potem dodaje odpowiednie krawędzie do szkicu.

3. Ostatecznie wykazujemy, że trzeci algorytm rzeczywiście konstruuje prawidłowy graf najkrótszych ścieżek G_{uv}. Oczywiście każda ścieżka nieprzechodząca przez zbiór punktów orientacyjnych może być skonstruowana przez przeszukiwanie G^- z użyciem dwukierunkowego BFS. Jeśli ścieżka przechodzi przez R, to odpowiednie wierzchołki z R znajdują się w S_{uv} i ścieżki je zawierające są wyliczane w ostatnim kroku algorytmu.

4.7 Złożoność czasowa (szkic)

W algorytmie etykietowania czas wykonania BFS jest $O(|E|)$, co daje łączną złożoność etykietowania $O(|R||E|)$. Przyjmujemy, że $|R| = O(1)$, np. $|R| = 20$. Złożoność szkicowania wynosi $O(|R|^4)$, ale może zostać zmniejszona do $O(|R|^2)$ przez wcześniejsze obliczenie i zapamiętanie najkrótszych ścieżek między parami punktów orientacyjnych na meta-grafie. Czas przeszukiwania to z kolei $O(|E| + |R||V|)$ lub też $O(|E^*| + |V|)$, gdzie $|E^*|$ jest liczbą krawędzi odchudzonego grafu.

4.8 Urównoleglenie

Łatwo pokazać, że dla danego zbioru punktów orientacyjnych R w G, istnieje dokładnie jeden system etykietowania $\mathcal{L}$ spełniający definicję, jego wybór jest więc oczywiście deterministyczny. Pozwala to łatwe urównoleglenie algorytmu etykietowania.

4.9 Złożoność pamięciowa

Autorzy nie przeprowadzają dokładnej analizy złożoności pamięciowej. Łatwo jednak zauważyć, że dodatkowo wykorzystana pamięć nie przekracza w asymptotyce tej koniecznej do reprezentacji grafu. Można więc założyć, że wynosi ona $O(|E|)$. Eksperymenty zdają się wykazywać w tym wypadku złożoność liniową.

4.10 Zależność od liczby punktów orientacyjnych

Generalnie dla grafów w których stopnie wierzchołków są zróżnicowane, wybór większej liczby punktów orientacyjnych (w szczególności tych o wysokim stopniu), skraca czas zapytań. Inaczej sprawa ma się w grafach o dość jednostajnych stopniach, gdzie zwiększony czas obliczania szkicu może pogorszyć ogólne rezultaty. Oczywiście większa liczba punktów orientacyjnych zwiększa wykorzystanie pamięci

5 Zakończenie

Dziękuję słuchaczom za uwagę i życzę miłego dnia.