

Kierunek: **Informatyka algorytmiczna (INA)**

PRACA DYPLOMOWA
MAGISTERSKA

**Zastosowanie szkiców danych w
analizie dużych grafów**

**Application of data sketches in the
analysis of large graphs**

Paweł Polerowicz

Opiekun pracy
dr inż. Jakub Lemiesz

Słowa kluczowe: TODO, TODO, TODO

Streszczenie

Polski

Słowa kluczowe: TODO, TODO, TODO

Abstract

English

Keywords: TODO, TODO, TODO

Spis treści

1. Wstęp	8
1.1. Struktura pracy	8
2. Opis problemu	9
2.1. Analiza grafów	9
2.2. Główne sposoby modelowania problemu	9
2.2.1. Strumienie grafów	9
2.2.2. Model półstrumieniowy TODO: Potwierdzić nazwę	9
2.3. Definicja formalna	9
3. Przegląd literatury	10
3.1. Streszczenia/Sketch synopses TODO: Potwierdzić nazewnictwo	10
3.2. MDL	10
3.3. Metody oparte na modyfikacji macierzy sąsiedztwa	11
3.4. Semi-Streaming model	13
3.5. Embeddings	13
4. Szkice danych	14
4.1. Definicja	14
5. Główny przedmiot pracy	15
5.1. Co nas najbardziej interesuje	15
5.2. Jaki jest ogólny pomysł	15
5.3. Jak to dokładnie zrobiliśmy	15
6. Analiza wyników	16
6.1. Architektura eksperymentów	16
6.2. Wyniki	16
6.3. Wnioski	16
Literatura	17

Spis rysunków

Spis tabel

Spis listingów

Skróty

ISO (ang. *International Standards Organization*)

Rozdział 1

Wstęp

TODO: Tutaj będzie bardzo ogólne wprowadzenie do pracy.

- Motywacja
- Krótki i "wysokopoziomowy" opis problemu
- Podsumowanie osiągnięć pracy

1.1. Struktura pracy

TODO: Kilka(dziesiąt) słów o strukturze i zawartości pracy. Omówienie po kolei rozdziałów i ewentualnych dodatków. Bardzo skrótowo, bo wszystko będziemy potem i tak rozwijać. Coś w jak niżej (do dopracowania).

Pierwszy rozdział stanowi niniejszy wstęp, przedstawiający ogólny zarys problematyki pracy i skrótowo podsumowujący jej wkład badawczy. W drugim rozdziale znajduje się opis problemu wraz z formalną definicją i przedstawieniem różnych jego wariantów. Przedmiotem trzeciego rozdziału jest przegląd literatury związanej z analizą wielkich grafów, z podziałem na zastosowane metodyki oraz tabelą ilustrującą porównanie znanych struktur i algorytmów. W czwartym rozdziale szczegółowo omówione zostały szkice danych, od ich formalnej definicji do bardziej praktycznych przykładów ich wykorzystanie, także w kontekście niniejszej pracy. Piąty rozdział zawiera właściwy opis tego, co zostało zrobione [TODO: przepisać nieco bardziej szczegółowo]. W rozdziale szóstym opisane zostały przeprowadzone eksperymenty, wraz z prezentacją wyników oraz wnioskami z nich płynącymi. Ostatni, siódmy rozdział, stanowi podsumowanie pracy. Zawarte zostały w nim ogólne konkluzje na temat pracy oraz możliwe kierunki dalszych badań. Pracy towarzyszy wykaz literatury oraz dodatek, zawierający opis dołączonej płyty CD [TODO: czy nadal załączamy płytę?] i instrukcję użytkowania części implementacyjnej.

Rozdział 2

Opis problemu

2.1. Analiza grafów

2.2. Główne sposoby modelowania problemu

2.2.1. Strumienie grafów

2.2.2. Model półstrumieniowy TODO: Potwierdzić nazwę

Model półstrumieniowy (ang. *semi-streaming model*)[5] jest wariantem klasycznego modelu strumieniowego, z dwiema głównymi różnicami. Po pierwsze, narzuca on konkretne ograniczenia na pamięć wykorzystywaną przez algorytm, najczęściej $O(|V|polylog(|V|))$, a więc dla gęstych grafów znacznie mniejszą niż rozmiar grafu. Po drugie, wejście może być skanowane wielokrotnie, zwykle stałą lub logarytmiczną liczbę razy. Model ten można uznać więc za rodzaj pomostu między klasyczną analizą grafów, w których dane znane są od początku i nie istnieją ograniczenia na dostęp do nich, a modelem strumieniowym, który nie pozwala na wielokrotne przeglądanie wcześniejszych krawędzi. Model ten jest często wybierany przez badaczy analizujących problemy grafowe takie, jak np. wyznaczanie najkrótszych ścieżek [4] lub minimalnego drzewa rozpinającego [1] przy rygorystycznych ograniczeniach pamięciowych.

2.3. Definicja formalna

Rozdział 3

Przegląd literatury

Analiza wielkich grafów, zwłaszcza w ostatnich latach, przeżywa ogromny rozwój, budząc zainteresowanie grup badaczy z całego świata. Postępy w tej dziedzinie są naturalną odpowiedzią na potrzebę przetwarzania coraz większych zbiorów danych. Badanie interakcji w sieciach społecznościowych, zarządzanie ruchem internetowym, czy monitorowanie ruchu samochodowego to tylko niektóre z kluczowych w dzisiejszej rzeczywistości zastosowań. Grafy dobrze sprawdzają się jako modele do reprezentowania złożonych relacji między encjami, dzięki czemu odgrywają kluczową rolę w przetwarzaniu i wydobywaniu wniosków ze strumieniowanych danych. Wybrane do tych celów algorytmy i metodologie w znacznym stopniu zależą od struktury badanych grafów, a także charakteru zapytań, które chcemy rozpatrywać. W zależności od wymagań dotyczących złożoności czasowej i pamięciowej, dokładności odpowiedzi, a także konkretnych informacji, na których zachowaniu nam zależy, inne metody mogą okazać się najlepszym wyborem. Przykładowo, odpowiedź na pytania o najkrótsze ścieżki między wierzchołkami może wymagać zapamiętania dodatkowych informacji o strukturze grafu, a więc potencjalnie użycia bardziej wyrafinowanego podejścia niż w przypadku zapytań wyłącznie o istnienie danej krawędzi.

W niniejszym przeglądzie literatury zagłębiamy się w sferę analizy dużych grafów ze szczególnym uwzględnieniem metod opartych na szkicach danych. Badamy ewoluujący krajobraz technik, algorytmów i aplikacji w tej dziedzinie, rzucając światło na metodologie stosowane w celu sprostania nieodłącznym wyzwaniom stawianym przez strumieniowe przesyłanie danych grafowych. Poprzez analizę najnowszych osiągnięć, staramy się zapewnić wgląd w znaczenie i potencjał analizy strumieni grafów w rozwiązywaniu złożonych zadań analitycznych w różnych dziedzinach, a także sformułować ogólne wnioski i wskazówki co do wyboru odpowiedniej metody do danego zastosowania.

3.1. Streszczenia/Sketch synopses **TODO: Potwierdzić nazewnictwo**

3.2. MDL

Graph summaries - Graph pattern matching over streams Graph stream algorithms Distributed graph systems

3.3. Metody oparte na modyfikacji macierzy sąsiedztwa

Jednym z najbardziej popularnych i być może najprostszym koncepcyjnie sposobem reprezentacji grafu jest macierz sąsiedztwa. Jej wiersze i kolumny odpowiadają poszczególnym wierzchołkom, a w komórkach przechowywane są wagi krawędzi pomiędzy nimi, o ile takowe krawędzie istnieją. W przypadku grafów nieważonych może to być np. wartość logiczna indykująca istnienie krawędzi lub ustalona stała. Ten sposób reprezentacji ma niewątpliwe zalety takie jak prostota implementacji i, przede wszystkim, stały czas dostępu do wag krawędzi. Z tego powodu

Jedną z pierwszych realizacji tej idei o przechowywaniu informacji o strumieniowanym grafie jest struktura TCM[8]. Ma ona postać macierzy o boku długości m , gdzie m jest pewną stałą. Podobnie jak w klasycznej macierzy sąsiedztwa, w jej komórkach składowane są wagi krawędzi. Zasadniczą różnicą jest natomiast sposób wyznaczania rzędu i kolumny odpowiadających danej parze wierzchołków. Są one bowiem wyznaczane przez wynik funkcji haszującej $H : V \rightarrow [1..m]$. Czas obliczania hasza jest stały, a co za tym idzie, złożoność czasowa zapytań i dodawania nowych krawędzi również. Teoretyczna złożoność pamięciowa także jest stała i wynosi $O(m^2)$. W praktycznych zastosowaniach wybór m zależy jednak często od liczby krawędzi i przejmuje się najczęściej m rzędu $O(\sqrt{|V|})$. Dokładność rezultatów zależy od rozmiaru macierzy i może być niska ze względu na kolizje haszy. Łatwo zauważyć, że jeśli m jest istotnie mniejsze od $|V|$ to może do nich dochodzić często, co powoduje traktowanie różnych krawędzi jako kolejnych instancji tego samego połączenia. Autorzy, świadomi tego ograniczenia, proponują zastosowanie kilku parami niezależnych funkcji haszujących i stworzenie na ich podstawie wielu szkiców grafu. Przykładowo, jeśli badaną zmienną jest suma wag kolejnych instancji krawędzi między danymi dwoma wierzchołkami, to algorytm może sprawdzić odpowiednie komórki dla wszystkich szkiców, a następnie zwrócić maksymalną wartość. Podejście to pozwala na analizę większych grafów niż w przypadku pojedynczego szkicu, ale ostatecznie nie rozwiązuje całkowicie problemu. Użyteczność struktury TCM w bazowej formie jest dyskusyjna, stanowi ona jednak punkt wyjściowy dla bardziej zaawansowanych rozwiązań.

Strukturą opartą na koncepcie podobnym do TCM jest *Graph Stream Sketch* (GSS)[6]. Celem autorów było stworzenie metody oferującej lepszą skalowalność dla wielkich grafów. Podobnie jak w TCM, funkcja haszująca mapuje zbiór wierzchołków na pewien mniejszy zbiór M -elementowy. Rozmiar macierzy jest natomiast równy m , $m < M$. Główną zmianą jest wprowadzenie dodatkowych cech opisujących wierzchołki. Na podstawie hasza $H(v)$ wyznaczany jest podpis wierzchołka $f(v)$ ($0 \leq f(v) < F$), gdzie $M = m \times F$ i $f(v) = H(v) \% F$, a także adres $h(v) = \lfloor \frac{H(v)}{F} \rfloor$. Adresy służą do wyznaczania rzędu i kolumny komórek. Komórki te mają one postać krotki lub, bardziej obrazowo, kubelka, w którym przechowywana jest para podpisów wierzchołków tworzących krawędź oraz kumulatywna waga krawędzi. Przechowywanie podpisów w komórkach pozwala zredukować ryzyko konfliktu haszy. Łatwo bowiem zauważyć, że nawet jeśli dwa różne wierzchołki mają taki sam adres, to istnieje duża szansa, że ich podpisy są różne. Z tego względu nowa krawędź jest dodawana do kubelka tylko w wypadku, gdy jest on pusty lub gdy istniejące w nim podpisy są zgodne z podpisami wierzchołków krawędzi tą tworzących. W przeciwnym przypadku jest ona zapisywana w dodatkowym buforze, mającym postać listy sąsiedztwa pełnych haszy. Pozwala on na dodawanie nowych krawędzi z niskim ryzykiem kolizji, nawet jeśli sama macierz jest już zapełniona. Należy natomiast zauważyć, że część macierzowa struktury jest bardziej efektywna czasowo, oferując stały czas odpowiedzi na zapytanie, podczas gdy dla bufora jest on liniowy względem liczby wierzchołków. Dokładność odpowiedzi w części macierzowej zależy od długości podpisów. Potencjalnym problemem GSS jest niskie wykorzystanie pamięci w macierzy. Przy kolizji adresów nowe krawędzie mogą trafiać do bufora, mimo, że w samej macierzy pozostaje wiele pustych komórek. Aby temu zaradzić, autorzy proponują haszowanie krzyżowe (*ang. square-hashing*). Zakłada ona obliczanie dla każdego wierzchołka sekwencji niezależnych adresów. Podczas wstawiania nowych krawę-

dzi algorytm sprawdza nie jedną komórkę macierzy, a kilka, zgodnie z sekwencją adresów i wybiera pierwszą spełniającą wymagania co do zgodności podpisów. Istnieje probabilistyczne ograniczenie na błąd względny zapytań postaci $Pr(\tilde{f}(s, d) - f(s, d)/\bar{w} > \delta) \leq \frac{|E|}{\delta m^2 4^T}$, gdzie $\tilde{f}(s, d)$ jest zwróconą sumą wag krawędzi (s, d) , $f(s, d)$ jej rzeczywistą wartością, \bar{w} średnią wagą krawędzi, a f długością podpisu.

Większość struktur służących podsumowywujących strumieniowane grafy nie przechowuje informacji o czasie wystąpienia krawędzi. Nie wspierają one więc zapytań z zakresem czasowym, a więc np., czy dana krawędź wystąpiła w zakresie $[t, t + L)$. Tego typu zapytania mogą być kluczowe np. w przypadku analizy danych dotyczących rozprzestrzeniania się wirusów (TODO: Citation needed). Problem ten podejmuje praca proponująca strukturę Horae[3]. W jej wypadku krawędź $e_i = (s_i, d_i, w_i, t_i)$ jest wstawiana do komórki o adresie $(h(s_i|\gamma(t_i)), h(d_i|\gamma(t_i)))$, gdzie $\gamma(t_i) = \lfloor \frac{t_i}{gl} \rfloor$ i gl jest długością przedziałów czasowych. Intuicyjnie, zapytanie o pojawienie się krawędzi w zakresie czasowym $[T_b, T_e]$ może być transformowane w sekwencję zapytań o pojedyncze zakresy, których wyniki są sumowane, a więc $Q([T_b, T_e]) = Q([T_b]), Q([T_{b+1}]), \dots, Q([T_e])$. Jednak dla takiego algorytmu złożoność czasowa jest liniowa względem liczby zakresów. Autorzy starają się poprawić ten aspekt, zauważając, iż przedział długości L może zostać zdekomponowany do co najwyżej $2 \log L$ podprzedziałów posiadających dwie szczególne cechy. Po pierwsze, wszystkie zakresy czasowe w danym podprzedziale mają wspólny prefiks binarny. Po drugie, prefiksy różnych podprzedziałów mają różne długości. Z tego względu Horae zapamiętuje $O(\log(T))$ identycznych skompresowanych macierzy, gdzie T jest liczbą rozróżnialnych zakresów czasowych. Każda z nich jest utożsamiana z jedną warstwą struktury. Warstwy odpowiadają z kolei różnym długościom prefiksów. Dzięki temu zamiast wykonywać liniową względem długości przedziału czasowego liczbę zapytań, wystarczy zdekomponować przedział na podprzedziały i na ich podstawie wykonać co najwyżej jedno zapytanie na warstwę.

Metody oparte na macierzach w większości przypadków nie czynią założeń co do struktury grafu. Takie ogólne podejście oczywiście zapewnia wysoką uniwersalność, jednak w niektórych przypadkach może być nieefektywne. Przykładowo, jeśli wierzchołki w grafie są mocno zróżnicowane pod względem stopnia, a więc bardziej obrazowo, da się wyróżnić obszary gęste i rzadkie w grafie, to kolizje haszy mogą zdarzać się często. Struktura Scube[2] używa probabilistycznego zliczania do identyfikacji wierzchołków wysokiego stopnia. Przeznaczane jest dla nich więcej kubeków w macierzy niż dla wierzchołków o niskich stopniach, co pozwala bardziej efektywnie zarządzać zapełnieniem macierzy.

Metody takie jak GSS czy Horae, choć często dają przyzwoite wyniki przy odpowiednim dobraniu parametrów do badanego grafu, to ostatecznie cierpią z uwagi na ograniczoną skalowalność. Jedną z prób odpowiedzi na ten problem jest struktura AUXO[7]. Korzysta ona z macierzy przechowujących podpisy wierzchołków, podobnie jak GSS. Jednak, zamiast wstawiać nadmiarowe krawędzie do bufora o liniowym czasie dostępu, AUXO wykorzystuje wiele macierzy ustawionych w strukturę drzewa. Konkretnie, jest to binarne lub czwórkowe drzewo prefiksowe, w którego strukturę zaszyte zostały prefiksy podpisów wierzchołków. W ten sposób na każdym kolejnym poziomie drzewa podpisy przechowywane w komórkach mogą być coraz krótsze, gdyż informacja ta jest wbudowana w kształt struktury. Pozwala to osiągnąć logarytmiczny względem liczby krawędzi czas odpowiedzi na zapytania. Warto zauważyć, że złożoność pamięciowa jest ograniczona przez długość podpisów, która wyznacza maksymalną głębokość drzewa. Niemniej jednak liczba możliwych do przetworzenia krawędzi jest eksponencjalna w stosunku do liczby bitów podpisu, więc stosunkowo łatwo można dobrać wystarczające wartości. W praktycznych zastosowaniach autorzy wskazują, nieco niefortunnie, na złożoność pamięciową zbliżoną asymptotycznie do $O(|E|(1 - \log(E)))$. Jak widać, AUXO osiąga efektywność pamięciową i skalowalność kosztem zwiększenia złożoności czasowej, co może być potencjalną wadą tego rozwiązania.

3.4. Semi-Streaming model

3.5. Embeddings

Rozdział 4

Szkice danych

4.1. Definicja

Rozdział 5

Główny przedmiot pracy

5.1. Co nas najbardziej interesuje

5.2. Jaki jest ogólny pomysł

5.3. Jak to dokładnie zrobiliśmy

Rozdział 6

Analiza wyników

6.1. Architektura eksperymentów

6.2. Wyniki

6.3. Wnioski

Literatura

- [1] K. J. Ahn, S. Guha, A. McGregor. Analyzing graph structure via linear measurements. *Proceedings of the Twenty-Third Annual ACM-SIAM Symposium on Discrete Algorithms*, Jan 2012.
- [2] M. Chen, R. Zhou, H. Chen, H. Jin. Scube: Efficient summarization for skewed graph streams. *2022 IEEE 42nd International Conference on Distributed Computing Systems (ICDCS)*, 2022.
- [3] M. Chen, R. Zhou, H. Chen, J. Xiao, H. Jin, B. Li. Horae: A graph stream summarization structure for efficient temporal range query. *2022 IEEE 38th International Conference on Data Engineering (ICDE)*, 2022.
- [4] M. Elkin, C. Trehan. Brief announcement: $(1+\epsilon)$ -approximate shortest paths in dynamic streams. *Proceedings of the 2022 ACM Symposium on Principles of Distributed Computing*, 2022.
- [5] J. Feigenbaum, S. Kannan, A. McGregor, S. Suri, J. Zhang. On graph problems in a semi-streaming model. *Theoretical Computer Science*, 348(2–3):207–216, 2005.
- [6] X. Gou, L. Zou, C. Zhao, T. Yang. Fast and accurate graph stream summarization. *2019 IEEE 35th International Conference on Data Engineering (ICDE)*, 2019.
- [7] Z. Jiang, H. Chen, H. Jin. Auxo: A scalable and efficient graph stream summarization structure. *Proceedings of the VLDB Endowment*, 16(6):1386–1398, 2023.
- [8] N. Tang, Q. Chen, P. Mitra. Graph stream summarization. *Proceedings of the 2016 International Conference on Management of Data*, 2016.