

Seminar 'Indizieren und Anfragen von Graphen in Datenbanken'

Maria Tsitiridou

Humboldt-Universität zu Berlin, Institut für Informatik, 10099 Berlin

{tsitirid}@informatik.hu-berlin.de

Zusammenfassung Die Arbeit beruht auf eine Veröffentlichung von James Cheng, Yiping Ke, Wilfred Ng und An Lu [2] mit dem Titel 'FG-Index: Towards Verification-free Query Processing on Graph Databases', in der eine ganz neue Graphindizierungstechnik vorgeschlagen wird.

Dabei wird ein verschachtelter invertierter Index erstellt, genannt der FG-Index, der eine Menge der häufigen Subgraphen (*Frequent Subgraphs* ($FG's$)) indiziert, um eine effektivere Anfrageverarbeitung zu erzielen.

Er wird im Paper mit dem bis dahin besten GraphIndex, den sogenannten gIndex [5], verglichen. Durch mehrere Experimente wird festgestellt, dass der FG-Index leistungsmäßig enorm besser abschneidet als der gIndex.

Inhaltsverzeichnis

Seminar 'Indizieren und Anfragen von Graphen in Datenbanken'	1
<i>Maria Tsitiridou</i>	
1 Einleitung	3
2 Vorbemerkungen	4
3 Formulierung des Problems	4
4 δ -Tolerance Closed Frequent Subgraphs (δ -TCFGs)	6
4.1 Häufige Subgraphen	6
4.2 Die Idee von δ -TCFGs	7
5 FG-Index	9
5.1 Index-Konstruktion	10
5.2 Anfrage-Verarbeitung	11
5.2.1 Anfrage q ist ein FG	11
5.2.2 Anfrage q ist kein FG	11
6 Performanz-Bewertung	11
7 Schlußbemerkungen	13

1 Einleitung

Graphen werden häufig benutzt, um die Beziehungen zwischen Objekten in verschiedenen Bereichen zu modellieren. In wissenschaftlichen Gebieten werden Graphen verwendet, um z.B. die Molekular-Strukturen von chemischen Verbindungen oder die Sozialen Netzwerke zwischen Menschen zu modellieren. Speziell im Gebiet der Informatik werden Graphen sehr häufig verwendet, wie z.B. die E-R-Diagramme im Datenbankdesign oder die graphischen Modelle in der künstlichen Intelligenz. Außerdem können die schnell zunehmende Zahl an Websites, sowie XML-Dokumente auch als Graphen dargestellt werden. Mit der zunehmenden Nutzung der Graph-Datenbanken wurde es immer wichtiger, aber auch anspruchsvoller, Graphanfragen effizient zu bearbeiten. Anfragen in Graphdatenbanken sind sehr zeitaufwändig, da sie viele Subgraph-Isomorphie-Tests erfordern, die bekanntlich ein NP-vollständiges Problem darstellen.

Früher haben manche effiziente Indizes durch Aussortieren von falschen Ergebnissen eine Kandidaten-Antwortmenge erstellt, die potentielle Antworten enthielt. Anschließend wurde eine Verifikation, in Form von Subgraph-Isomorphie-Tests, an jedem Graphen dieser Menge durchgeführt, um festzustellen, ob es sich tatsächlich um eine Antwort handelte. Wenn die Kandidatenmenge zu groß war, war die Verifikation sehr zeitaufwändig.

Die neue Indizierungstechnik vom FG-Index reduziert die Anzahl der Subgraph-Isomorphie-Tests, was für größere Anfrage-Effizienz sorgt. Wenn es sich bei der Anfrage um einen häufigen Subgraphen handelt, ist der FG-Index in der Lage die Antwortmenge sofort zu liefern, ohne eine Kandidaten-Verifikation durchführen zu müssen. Andernfalls, wenn es sich um einen nicht häufigen Subgraphen handelt, was bedeutet, dass er in nicht so vielen Graphen in der Datenbank enthalten ist, ist die Kandidatenmenge sehr klein und daher auch die Anzahl der Subgraph-Isomorphie-Tests. Außerdem verwendete man bei der Konstruktion des FG-Index eine neue Auffassung der Idee der δ -Tolerance Closed Frequent Graphs ($\delta - TCFGs$), einer Kompressionstechnik, durch die man die große Menge der häufigen Subgraphen komprimieren kann, damit der Index nicht zu groß für den Hauptspeicher wird, was wiederum die Zugriffszeiten verkürzt und folglich die Performanz erhöht.

Durch eine umfangreiche Menge von Experimenten konnte das Team von J.Cheng bestätigen, dass der FG-Index den bis dahin besten GraphIndex, gIndex[5], leistungsmäßig in Indexkonstruktion und Verarbeitung der Anfrage enorm übertrifft. Insbesondere ist die Anfrage-Verarbeitung unter Verwendung vom FG-Index für einen weiten Bereich von Anfragen erheblich schneller als die unter Verwendung von gIndex. Die Ergebnisse bestätigten auch, dass die Idee der $\delta - TCFGs$ bei der Steuerung der Größe der Index-Speicherbelegung sehr effektiv ist. Zusätzlich scheint der FG-Index eine gute Skalierbarkeit in Bezug auf die Datenbank-Größe, die Graphen-Größe und Graphen-Dichte zu haben.

2 Vorbemerkungen

Für das Verständnis der weiteren Arbeit führen wir zuerst einige Begriffe ein.

Definition 1 Ein *Graph* g ist ein 4-Tupel (V, E, L, l) , wobei V die Menge der Knoten, E die Kantenmenge, L die Menge der Labels und l eine labelling-Funktion ist, die jedem Knoten und jeder Kante ein Label aus L zuordnet.

Definition 2 Die *Größe*, "size(g)", des Graphen g ist definiert durch $size(g) = |E(g)|$

Bsp.: Graph f_1 in Abbildung 1 ist ein Graph, der Label a besitzt und die Größe 1 hat.

Definition 3 Eine *eindeutige Kante* in G ist ein 3-Tupel (l_u, l_e, l_v) , wobei G eine Menge von Graphen, l_e das Label einer Kante (u, v) in einem Graphen und l_u und l_v die Labels von u und v in g sind.

Definition 4 Sei e eine eindeutige Kante, $count(e, g)$ ist die Anzahl der Vorkommnisse von e in g .

Bsp.: Graph f_5 in Abbildung 1 zeigt einen Graphen mit 3 Kanten a , 2 Kanten c und 1 Kante b , wobei a , b , c eindeutige Kanten sind.

Definition 5 Gegeben seien zwei Graphen, $g = (V, E, L, l)$ und $g' = (V', E', L', l')$. Eine *Subgraph-Isomorphie* von g nach g' ist eine injektive Funktion $f : V \rightarrow V'$ so, dass $\forall (u, v) \text{ aus } E, (f(u), f(v)) \text{ aus } E', l(u) = l'(f(u)), l(v) = l'(f(v))$ und $l(u, v) = l'(f(u), f(v))$.

Definition 6 Ein Graph g heißt *Subgraph* von einem anderen Graphen $g', g \subseteq g'$ (oder $g' \supseteq g$), wenn eine Subgraph-Isomorphie von g nach g' existiert.

Bsp.: Graph f_4 in Abbildung 1 ist ein Subgraph von f_5 .

Definition 7 Ein *häufiger Subgraph* ist ein Graph, der in mindestens $\sigma |D|$ vielen Graphen enthalten ist, wobei $\sigma (0 \leq \sigma \leq 1)$ ein benutzerdefinierter Grenzwert ist.

Im Folgenden, wenn wir von Graphen sprechen, sind immer ungerichtete gelabelte Graphen gemeint.

3 Formulierung des Problems

Sei $D = \{g_1, g_2, \dots, g_N\}$ eine Graphdatenbank, die N Graphen enthält. Eine typische Graphanfrage q lautet wie folgt: Sei q ein Graph, finde alle Graphen in der Datenbank D , die q enthalten. Ab jetzt nennen wir eine Graphanfrage einfach eine Anfrage. Unsere

Aufgabe besteht darin, die Antwortmenge $D_q = \{g : g \in D \mid q \subseteq g\}$ der Anfrage q zu berechnen. Ein Beispiel für eine Graphdatenbank wird in Abbildung 1 gezeigt, wobei a, b, c eindeutige Kanten sind. Abbildung 2 zeigt ein Beispiel für eine Anfrage q . In unserem Beispiel besteht die Antwortmenge von q nur aus dem Graph f_6 , d.h. $D_q = \{f_6\}$, da q nur in f_6 enthalten ist. Da logischerweise Graphdatenbanken in der Praxis erheblich größer als unsere Beispieldatenbank sind, ist die Bearbeitung der Anfrage durch sequentielle Überprüfung in der Datenbank undurchführbar, da das Testen auf Subgraph-Isomorphie bekanntlich ein NP-vollständiges Problem ist.

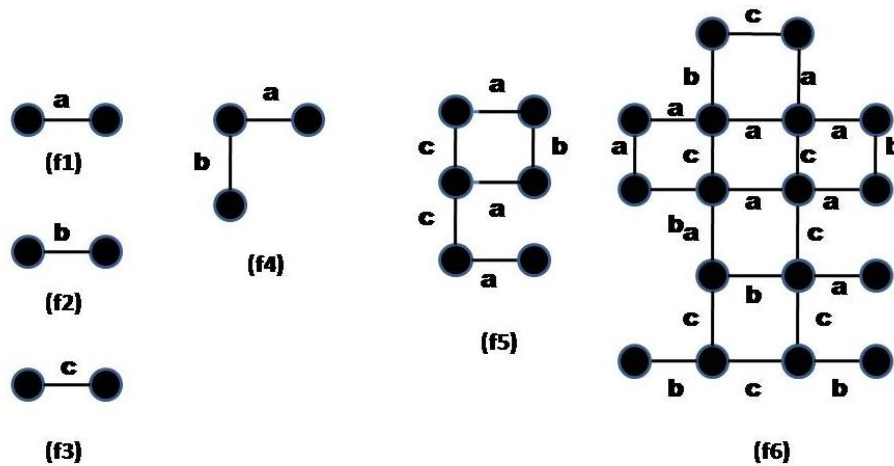


Abbildung 1. Beispiel einer Graphdatenbank.

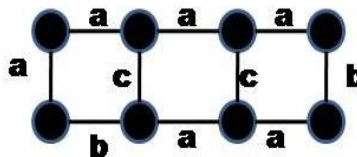


Abbildung 2. Beispiel einer Graphanfrage q .

In der Vergangenheit sind mehrere effektive Indizes in Graphdatenbanken vorgeschlagen worden, um die Performanz der Anfrageverarbeitung zu verbessern. Die Anfrageverarbeitung unter Verwendung dieser Indizes wird in zwei grundlegende Schritte ausgeführt:

1. Das Aussortieren von einem Teil von falschen Ergebnissen, um eine Menge mit potentiellen Antworten, den sogenannten Kandidaten C_q , zu erhalten.
2. Die Kandidaten-Verifikation, wobei für jeden Graphen q' aus der Kandidatenmenge C_q per Subgraph-Isomorphie-Test überprüft wird, ob es sich um einen Übergraphen von q handelt.

Da die Kandidatenmenge C_q gewöhnlich sehr viel kleiner als die gesamte Graphdatenbank ist, ist die Anfrageverarbeitung unter Verwendung des Indizes bedeutsam effizienter als die Methode der sequentiellen Überprüfung. Aufgrund der hohen Komplexität der Subgraph-Isomorphie-Tests ist die Kandidaten-Verifikation immer noch sehr teuer, da sie Größe des Kandidaten mindestens die der tatsächlichen Antwortmenge ist. Die Anfrage-Performanz hat sich mit der Zeit verbessert, da sich die Anzahl der Subgraph-Isomorphie-Tests reduziert hat. Man fragt sich, ob man die teure Kandidaten-Verifikation ganz umgehen könnte. In [2] wird eine vielversprechende Antwort darauf gegeben. Es wird nämlich eine ganz neue Indizierungstechnik vorgeschlagen, die einen verschachtelten invertierten Index konstruiert, den sogenannten FG-Index, der nur häufige Subgraphen aus D enthält. Allerdings ist die Verifikation der Kandidaten immer noch unvermeidbar, wenn der Umfang von C_q groß ist.

Abgesehen von der Kandidaten-Verifikation ist die Größe des Indizes ebenso wichtig für die Effizienz der Anfrageverarbeitung. Die Tatsache, dass die Menge der häufigen Subgraphen für ein kleines σ sehr groß sein könnte, stellt für den FG-Index [2] eine Herausforderung dar. In dem Fall könnte der Index, der aus den häufigen Subgraphen gebildet werden soll, zu groß sein, um in den Hauptspeicher gehalten zu werden. Folglich würde die Anfrage-Performanz sinken, da die Verarbeitung häufige Zugriffe auf die Festplatte erfordern würde.

Inspiziert von der Arbeit über das Komprimieren der Menge von häufigen Mustern in [1, 3], wird in [2] eine neue Auffassung dieser Idee vorgeschlagen, die der δ -Tolerance Closed Frequent Subgraphs (δ -TCFGs), um die Menge der häufigen Subgraphen zu komprimieren. Die Idee der δ -TCFGs erlaubt uns, die Größe des FG-Index über einen Parameter δ flexibel einzustellen.

4 δ -Tolerance Closed Frequent Subgraphs (δ -TCFGs)

4.1 Häufige Subgraphen

Sei D eine Graphdatenbank und g ein Graph aus D , die Häufigkeit von g ist definiert als

$$freq(g) = |\{g' : g' \in D, g' \supseteq g\}|. \quad (1)$$

Sei F die Menge aller häufigen Subgraphen (FGs) gewonnen aus D . In unserem Beispiel in Abbildung 1, gilt: $freq(f_1) = 14$, $freq(f_2) = 9$, $freq(f_3) = 10$, $freq(f_4) = 9$, $freq(f_5) = 2$, $freq(f_6) = 1$.

- Ein Graph g heißt *maximaler häufiger Subgraph (MFG)* [4], wenn $g \in F$ und $\neg \exists g' \in F$ so, dass $g' \supset g$.
- Ein Graph g heißt *abgeschlossener häufiger Subgraph (CFG)* [6], wenn $g \in F$, $\neg \exists g' \in F$ so, dass $g' \supset g$ und $freq(g') = freq(g)$.

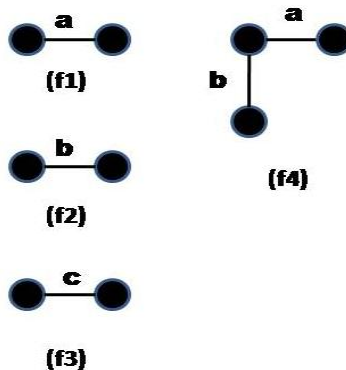


Abbildung 3. Häufige Subgraphen in D .

Abbildung 3 zeigt die Menge F der häufigen Subgraphen unserer Beispieldatenbank aus Abbildung 1. Unter den FGs ist f_4 ein MFG, da er keinen echten Übergraphen hat. Alle FGs außer f_2 sind CFGs. Der FG f_2 ist kein CFG, da $f_4 \supset f_2$ und $freq(f_4) = freq(f_2)$.

4.2 Die Idee von δ – $TCFGs$

In [2] wird der FG-Index vorgestellt, der häufige Subgraphen indiziert. Die Herausforderung besteht darin, wie man F indizieren kann, wenn F sehr *groß* ist. Um dieses Problem zu behandeln wurde die Idee der δ -Tolerance Closed Frequent Subgraphs (δ – $TCFGs$) [2, 1, 3] vorgeschlagen. Die Menge der δ – $TCFGs$, angegeben als T , ist eine komprimierte Darstellung von F . Jeder $g \in T$ steht in Zusammenhang mit einer Menge von FGs, die Subgraphen von g sind. Die Größe von T ist wesentlich kleiner als die von F . Folglich sind wir in der Lage einen Index, basierend auf T , zu konstruieren, der in dem Hauptspeicher gehalten werden kann. Zusätzlich werden verschachtelte Indizes für jeden $g \in T$ konstruiert und jeder von denen besteht jeweils aus der Menge der FGs, die in Zusammenhang zu dem entsprechenden $g \in T$ stehen. Diese verschachtelten Indizes bestehen aus der Mehrheit der FGs und werden aufgrund ihrer Größe auf der Festplatte und nicht im Hauptspeicher gespeichert.

Ein δ – $TCFG$ wird formell wie folgt definiert:

Definition 8 Ein häufiger Subgraph g ist ein $\delta - TCFG$, wenn folgendes gilt:

$$\neg \exists g' \in F : g' \supset g \text{ und } freq(g') \geq (1 - \delta) freq(g)$$

wobei δ ein benutzerdefinierter Häufigkeitsfaktor mit $0 \leq \delta \leq 1$ ist.

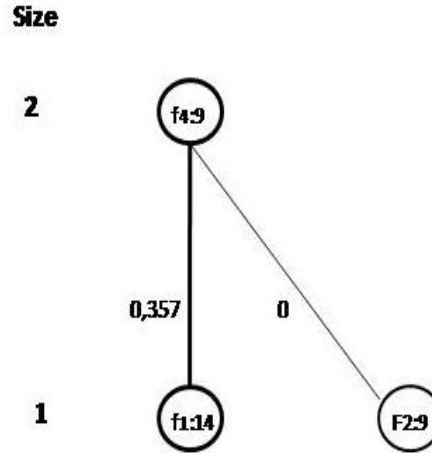


Abbildung 4. Häufige Subgraphen in D und ihre Häufigkeiten.

Beispiel 1 Man betrachte die 3 FGs in Abbildung 4. Die Zahl an jeder Kante e ist wie folgt berechnet worden: $d_e = (1 - freq(f_i) / freq(f_j))$, wobei f_i der kleinste zugehörige Übergraph von f_j ist, der auch die größte Häufigkeit hat. Wenn $d_e \geq \delta$ gilt, ist f_j laut Definition 8 ein $\delta - TCFG$, andernfalls ist f_j kein $\delta - TCFG$. Sei $\delta = 0,3$, dann ist die Menge der $(0,3 - TCFGs)$, $T_{0,3} = \{f_1, f_4\}$. In der Abbildung 4 sind das die Knoten, die fett gezeichnet sind. f_1 ist ein $(0,3 - TCFG)$, da die Bedingung $d_e \geq \delta$ aus Definition 8 erfüllt ist, d.h. weil $0,357 \geq 0,3$ ist. f_2 ist kein $(0,3 - TCFG)$, da $0 < 0,3$ gilt. Der Einfachheit wegen werden wir ab jetzt statt T_δ die einfachere Notation T benutzen, wenn δ im Kontext klar ist.

Da eine Anfrage aus $(F - T)$ sein kann, definieren wir die Verbindung zwischen den Graphen aus T und diesen aus $(F - T)$. Sei G eine Menge von Graphen und $\mathbb{N} = \{1, 2, 3, \dots\}$ die Menge der natürlichen Zahlen. Sei $h : G \rightarrow \mathbb{N}$ eine injektive Funktion, die jedem Graphen $g \in G$ eine eindeutige ID $n \in \mathbb{N}$ zuordnet. Wir definieren eine totale Ordnung \preceq in G wie folgt:

Definition 9 (*Graphenmenge – Ordnung*)

Seien $g_1, g_2 \in G$, es gilt $g_1 \preceq g_2$, wenn eine der folgenden drei Aussagen wahr ist:

1. $size(g_1) < size(g_2)$
2. $size(g_1) = size(g_2)$ und $freq(g_1) > freq(g_2)$
3. $size(g_1) = size(g_2)$, $freq(g_1) = freq(g_2)$ und $h(g_1) \leq h(g_2)$

Definition 10 (Nächster $\delta - TCFG$ Übergraph)

Seien $g_t \in T$ und $g \in (F - T)$. g_t heißt *nächster $\delta - TCFG$ Übergraph* von g , wenn

$$g_t \supset g \text{ und } \neg \exists g'_t \in T \text{ so, dass } g'_t \supset g \text{ und } g'_t \prec g_t.$$

Definition 11 (Closure von $\delta - TCFGs$)

Sei $g_t \in T$. *Closure* von g_t wird wie folgt definiert:

$$CLOS(g_t) = \{g : g_t \text{ ist der nächste } \delta - TCFG \text{ Übergraph von } g\}$$

Beispiel 2 Auf den Abbildungen 3 und 4 ist die Menge der FGs nach der definierten Ordnung \preceq geordnet. Wenn $\delta = 0, 3$ ist, ist f_4 der nächste $\delta - TCFG$ Übergraph von f_2 ; mit anderen Worten ist $CLOS(f_4) = \{f_2\}$, was wiederum bedeutet, dass f_4 in den invertierten Index, der im Hauptspeicher bleibt, aufgenommen wird, während f_2 in dem verschachtelten Index von f_4 liegt, der auf der Festplatte gespeichert wird.

5 FG-Index

Der FG-Index ist ein invertierter Graph-Index, der wie folgt beschrieben ist:

Sei G eine Menge von Graphen. Ein invertierter Graph-Index (IGI), erstellt aus G besteht aus den folgenden Komponenten:

- Ein Graph-Array (GA), in dem die Graphmenge G gespeichert wird.
Sei $GA[i]$ der i -te Eintrag in GA . Dem Graphen, der in $GA[i]$ gespeichert ist, wird eine ID i zugeordnet. Wenn G eine Menge von $\delta - TCFGs$ ist, enthält jedes $GA[i]$ auch die Speicheradresse von dem verschachtelten IGI, der aus der Menge $CLOS(g)$ gebildet ist, wobei g der an Stelle $GA[i]$ gespeicherte Graph ist.
- Ein Array, genannt Edge-Array (EA), in dem die Menge der eindeutigen Kanten in G gespeichert wird.
- Jede eindeutige Kante e in EA ist mit einer Liste von ID's von Graphen in G verknüpft, die die Kante enthalten.

Die ID's sind nach der Graphengröße weiter klassifiziert, um effizientere Suche zu erzielen. Die Zahl im linken Teil des ID-Eintrags gibt an, wie oft die Kante in dem entsprechenden Graphen vorkommt und die Zahl in seinem rechten Teil die Graph-ID im (GA).

Im rechten Teil des ID-Eintrags können mehrere ID's aufgelistet sein, wenn zum Beispiel die Kante e in mehreren Graphen der Größe n gleich oft vorkommt, sowie es auch möglich ist, dass der Size- n -Eintrag der Kante mehrere Arrays haben kann, wenn zum Beispiel die Kante in mehreren Graphen der Größe n unterschiedlich oft vorkommt.

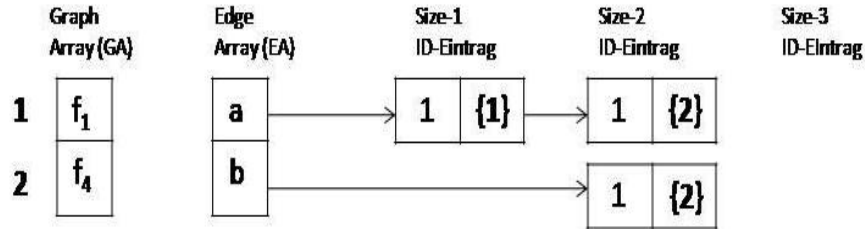


Abbildung 5. Invertierter Graph-Index vom Beispiel 3

Beispiel 3 Wir beziehen uns auf die FGs in Abbildung 3 und 4. Sei $\delta = 0,3$, dann ist $T = \{f_1, f_4\}$. Abbildung 5 zeigt den entsprechenden IGI, der basierend auf T konstruiert ist. Zum Beispiel hat die Kante a zwei ID-Einträge, einen Size-1 und einen Size-2 ID-Eintrag. Der Size-1 ID-Eintrag sagt also aus, dass die Kante a im Graph mit der $ID = \{1\}$, d.h. in f_1 , einmal vorkommt und der Size-2 ID-Eintrag entsprechend, dass sie im Graph mit der $ID = \{2\}$, also f_4 , auch einmal vorkommt.

5.1 Index-Konstruktion

Der FG-Index besteht aus den zwei folgenden Teilen: dem Core-FG-Index und dem Edge-Index. Zuerst wird ein invertierter Index konstruiert, der aus T gebildet wird und der im Hauptspeicher gespeichert wird. Danach wird für jeden $\delta - TCFG \in T$ ein invertierter Index aus dem Closure des entsprechenden $\delta - TCFG$, konstruiert. Auf dieselbe Weise, wenn das Closure sehr groß ist, kann eine lokale Menge von $(\delta - TCFGs)$ aus den FGs des Closures berechnet und ein weiterer verschachtelter invertierter Index gebildet werden. Der Core-FG-Index besteht aus dem invertierten Index im Hauptspeicher und aus allen verschachtelten invertierten Indizes auf der Festplatte. Der Core-FG-Index deckt nur die Anfragen ab, die häufige Subgraphen sind. Um sicherzustellen, dass jede Anfrage beantwortet werden kann, wurde zum FG-Index auch der Edge-Index hinzugefügt, der alle nicht häufigen eindeutigen Kanten enthält. Jede Kante e aus dem Edge-Index wird mit der Menge D_e der Graphen, die die Kante e enthalten, verknüpft. Der Algorithmus BuildIndex in [2], der für die Index-Konstruktion zuständig ist, erhält als Input eine Graphdatenbank D , die Menge der FGs F und den Häufigkeits-Toleranz-Faktor δ und besteht aus den folgenden drei Teilen:

1. Die Berechnung und das Sortieren von T
2. Die Konstruktion vom Core-FG-Index und
3. Die Erstellung vom Edge-Index.

Als Output des Algorithmus werden der Core-FG-Index und der Edge-Index ermittelt.

5.2 Anfrage-Verarbeitung

Die Verarbeitung einer Anfrage q kann unter Verwendung des FG-Index in zwei Fälle klassifiziert werden ($q \in F$) und ($q \notin F$)

5.2.1 Anfrage q ist ein FG

Wenn q ein häufiger Subgraph ist, wird der Algorithmus FG-Query aus [2] eingesetzt, der als Input den Core-FG-Index und die Anfrage q erhält und als Output D_q , die Menge der Übergraphen von q , zurückgibt. Der Algorithmus FG-Query verarbeitet nur diejenigen Graphen, die alle eindeutige Kanten von q enthalten. Er beginnt mit den Graphen mit der gleichen Größe wie q , bis ein Übergraph von q gefunden ist. Genauer:

- Zuerst wird der Hash-Index verwendet, um die Menge der Kanten von q im FG-Index zu finden.
- Der Durchschnitt von den Graph-ID-Arrays, die mit den Kanten verknüpft sind, wird gebildet.
- Der erste Übergraph, den wir über den Durchschnitt erhalten, ist entweder q oder q 's nächster $\delta - TCFG$ Übergraph.
- Wenn q immer noch nicht gefunden wurde, wird auf den verschachtelten IGI-Index von q 's nächsten $\delta - TCFG$ Übergraphen zugegriffen.
- Die Schritte werden rekursiv wiederholt, bis q gefunden wird.

5.2.2 Anfrage q ist kein FG

Wenn q kein häufiger Subgraph ist, gibt FG-Query aus [2] kein Ergebnis zurück. In diesem Fall wird der Algorithmus IFG-Query(*siehe*[2]) aufgerufen, der im Gegensatz zum FG-Query auch den Edge-Index als Input erhält. Genauer:

- Der FG-Index wird verwendet, um eine Menge G_s von maximalen Subgraphen von q zu finden.
- Der Edge-Index wird verwendet, um eine Menge G_e der nicht-häufigen Kanten von q zu finden.
- Die Kandidaten-Menge C_q von q wird über den Durchschnitt der Antwortmengen aus den Graphen in G_s und G_e gebildet.
- Für jeden Kandidaten aus C_q wird überprüft, ob er ein Übergraph von q ist.

Zusammenfassend, ist q ein FG, so müssen nur die Graphen zurückgegeben werden, ohne einen teuren Subgraph-Isomorphie-Test. Ist q allerdings kein FG, müssen alle Graphen in C_q durch einen Subgraph-Isomorphie-Test überprüft werden. Die Kandidatenmenge C_q ist aber häufig nicht groß, wie Kapitel 6 zeigt.

6 Performanz-Bewertung

In [2] wurde die Effizienz der Index-Konstruktion und der Anfrage-Verarbeitung vom FG-Index geprüft und mit der vom gIndex[5] verglichen. Das Team von James Cheng

hat in [2] für seine Experimente einen realen Datensatz verwendet, der auch bei der Bewertung von gIndex aus [5] verwendet worden war. Bei der realen Datenmenge handelte es sich um 43K Graphen einer Menge von chemischen Molekülen aus der AIDS-Forschung.

Weitere Experimente wurden auch mit künstlichen Datenmengen durchgeführt, um die Skalierbarkeit des FG-Index bzgl. der Datenbankgröße, der Graphengröße und der Graphendichte zu testen. Die Werte variierten für die Datenbankgröße von 10K bis 100K Graphen, für die Graphengröße von 20 bis 100 Kanten und für die Graphendichte zwischen 0.1, 0.15 und 0.2. Die Einstellungen für den gIndex sind die gleichen wie in [5]. Dabei wird die Auswirkung der Werte von σ und δ auf die Performanz vom FG-Index beobachtet. Tabelle 1 zeigt die Zeit der Index-Konstruktion und den Speicherverbrauch für den AIDS-Datensatz, wobei die Zeit vom FG-Index wie folgt zusammengestellt wird: FG-Gewinnungszeit + FG-Index-Konstruktionszeit. In der Tabelle 1 ist $|T|$ die Anzahl der ($\delta - TCFGs$), die im FG-Index[2] indiziert sind und $|F_d|$ die Anzahl der unterschiedlichen FG's, die im gIndex[5] indiziert sind. Die Konstruktions-

	FG-Index ($\sigma = 0.1$)	FG-Index ($\sigma = 0.01$)	gIndex
Time(sec)	10.03(10 + 0.03)	1085(627 + 458)	217
Memory (MB)	2	95	107
$ T $ or $ F_d $	195	21893	3276

Tabelle 1. Index-Konstruktions-Performanz für AIDS, Werte aus [2]

zeit vom FG-Index für AIDS ist bei $\sigma = 0.1$ 20 mal kleiner als die von gIndex, aber 5 mal größer bei $\sigma = 0.01$. Diese große Abweichung in der Konstruktionsperformanz vom FG-Index besteht, weil die Anzahl der FG's bei $\sigma = 0.1$ wesentlich kleiner ist als bei $\sigma = 0.01$ nämlich 455 vs 59120. Demzufolge ist die Anzahl der indizierten Graphen und der Speicherverbrauch auch wesentlich kleiner als bei $\sigma = 0.01$.

Für die Bewertung der Anfrage-Performanz wurden die sechs Anfragemengen $Q_4, Q_8, Q_{12}, Q_{16}, Q_{20}$ und Q_{24} verwendet, wobei jede Anfragemenge Q_i 1000 Anfragen enthält und jede Anfrage in Q_i aus i Kanten besteht.

Abbildung 6 zeigt die durchschnittliche Antwortzeit der Verarbeitung einer Anfrage in jedem Q_i . Das Ergebnis zeigt, dass die Anfrage-Verarbeitung bei beiden $\sigma = 0.1$ und $\sigma = 0.01$ unter Einsatz vom FG-Index schneller ist als unter Einsatz vom gIndex. Bei kleinen Anfragen, z.B. Q_4 , sogar wesentlich schneller. Der FG-Index schneidet für Q_4 am besten ab, da mehr Anfragen FG's sind und weniger Kandidaten-Verifikationen durchgeführt werden müssen als für andere Q_i 's. Im Gegenteil schneidet der gIndex[5] für Q_4 am schlechtesten ab, da jede Anfrage $q \in Q_4$ eine größere Antwortmenge D_q hat, was mehr Kandidaten-Verifikationen bedeutet. Die bessere Performanz des FG-Index, auch bei größeren Anfragen, liegt an der kleineren Anzahl der Kandidaten-Verifikationen, die erforderlich sind.

Auf Abbildung 7 kann man erkennen, dass der FG-Index auch bei unterschiedlichen Anfragegrößen einen erheblich kleineren und stabileren Speicherverbrauch hat

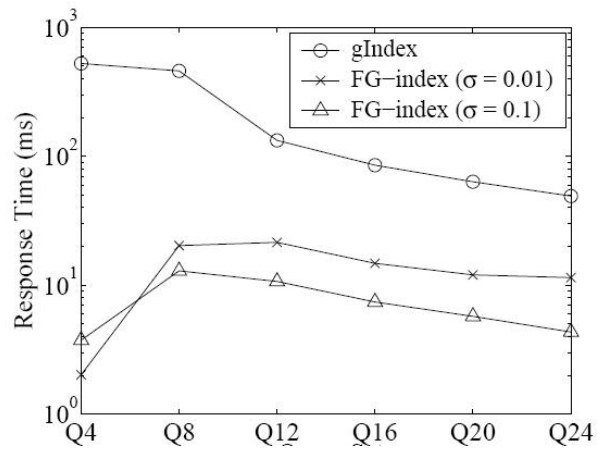


Abbildung 6. Antwortzeit bei unterschiedlichen Anfragegrößen, Abbildung aus [2]

als der gIndex. Der maximale Speicherverbrauch der Anfrage-Verarbeitung liegt unter Verwendung des FG-Index mit $\sigma = 0.1$ und $\sigma = 0.01$ für alle Q_i 's circa bei 1,8 MB bzw. 13 MB, während der vom gIndex durchschnittlich 30 MB beträgt.

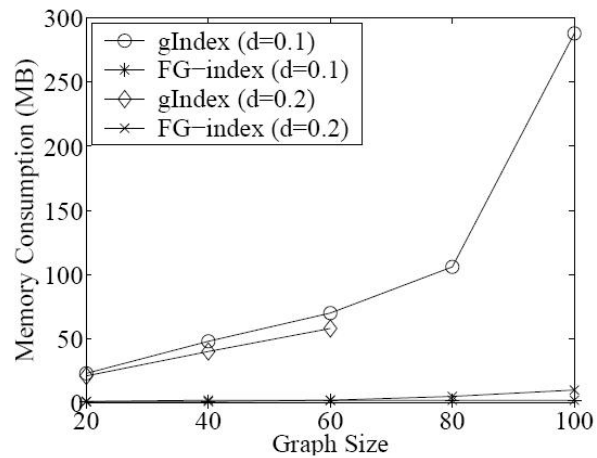


Abbildung 7. Speicherverbrauch bei unterschiedlichen Anfragegrößen, Abbildung aus [2]

7 Schlußbemerkungen

In [2] wurde der FG-Index vorgestellt, der ein effektiver Index ist, um effiziente Verarbeitung bei Graphen-Datenbanken zu unterstützen. Ein charakteristisches Merkmal

vom FG-Index ist, dass Anfragen, die häufige Subgraphen sind, ohne Durchführung von Kandidaten-Verifikation beantwortet werden können, während Anfragen die keine häufige Subgraphen sind, mit einer minimalen Anzahl von Kandidaten-Verifikationen beantwortet werden können. In [2] wurde im Thema Graph-Indizierung eine enorme Verbesserung über die Leistung der bisherigen Graph-Indizes hinaus erreicht, bei denen, im Gegensatz zum FG-Index, Kandidaten-Verifikation für jede Anfrage durchgeführt wird. Da die Menge der häufigen Subgraphen (FGs) sehr groß sein kann, wurde eine neue Komprimierungstechnik vorgestellt, die eine große Menge von FGs in eine kleine Menge von stellvertretenden FGs, genannt $\delta - TCFGs$, effektiv komprimiert. Experimentelle Ergebnisse des Teams von J.Cheng bestätigten die Effektivität der ($\delta - TCFGs$)-Technik im Komprimieren der Menge der FGs, sowie die enorme Effektivität des FG-Index gegenüber dem gIndex[5] in Index-Konstruktion und Anfrage-Verarbeitung. Insbesondere wurde gezeigt, dass nicht nur für Anfragen, die FGs sind, sondern auch für Anfragen, die keine FGs sind, die Anfrage-Verarbeitung unter Verwendung vom FG-Index sehr viel schneller ist als unter Verwendung vom gIndex[5].

Literatur

1. D. Xin, J. Han, X. Yan, and H. Cheng. Mining Compressed Frequent-Pattern Sets. In *Proceedings of the 31st International Conference on Very Large Data Bases (VLDB)*, pages 709–720. ACM Press, 2005.
2. J. Cheng, Y. Ke, W. Ng, and A. Lu. FG-Index: Towards Verification-Free Query Processing on Graph Databases. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*, pages 857–872. ACM Press, 2007.
3. J. Cheng, Y. Ke, and W. Ng. δ -Tolerance Closed Frequent Itemsets. In *Proceedings of the Sixth International Conference on Data Mining (ICDM)*, pages 139–148. IEEE Computer Society, 2006.
4. J. Huan, W. Wang, J. Prins, and J. Yang. SPIN: mining maximal frequent subgraphs from graph databases. In *Proceedings of the tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, pages 581–586. ACM Press, 2004.
5. X. Yan, P. S. Yu, and J. Han. Graph indexing based on discriminative frequent structure analysis. *ACM Trans. Database Syst.*, 30(4):960–993, 2005.
6. X. Yan and J. Han. CloseGraph: mining closed frequent graph patterns. In *Proceedings of the ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, pages 286–295. ACM Press, 2003.

Selbstständigkeitserklärung

Hiermit erkläre ich, dass ich die Seminararbeit selbstständig und nur unter Zuhilfenahme der angegebenen Quellen angefertigt habe.

Berlin, 12. April 2008