Exposé zur Studienarbeit: Implementierung der Indizierung von Graphen nach Agrawal

André Koschmieder

September 2007

1 Überblick

Graphen können die verschiedensten realen Bedeutungen haben, sei es eine Modellierung von Pfaden bei der Wegsuche, eine *IS-A* Beziehung von verschiedenen Typen, eine Klassenhierarchie o.Ä. Diese Graphen können sehr groß werden; so werden bei KEGG [4] Graphen mit mehreren zehntausend Knoten und Kanten behandelt. Auch bei solchen Graphen sollen Anfragen nach der Erreichbarkeit effizient beantwortet werden können, am besten in konstanter Zeit. Dafür ist eine Indizierung des Graphen nötig, bei der wichtig ist, dass die Erstellung des Index in der Praxis schneller als mit quadratischer Komplexität bzgl. der Knotenzahl erfolgt, der Speicherverbrauch möglichst gering ist, sowie effiziente Updatestrategien existieren.

Agrawal et al. schlagen in [2] als Index eine komprimierte transitive Hülle vor, welche die geforderten Eigenschaften besitzen soll.

Ich werde zunächst die Ideen von Agrawal darlegen, und anschließend auf die Ziele dieser Studienarbeit eingehen.

2 Komprimierte Transitive Hülle

Um Anfragen nach der Erreichbarkeit im Graphen möglichst effizient beantworten zu können, sollte zum Anfragezeitpunkt keine Wegsuche im Graphen stattfinden, sondern es sollte bereits bekannt sein, ob ein Pfad existiert. Um dies zu ermöglichen, muss ein Index erstellt werden, in dem zu jedem Knoten gespeichert wird, welche Knoten von ihm aus erreichbar sind (transitive Hülle). Da diese Hülle wegen des potentiell quadratischen Speicherbedarfs nicht in der Form von Adjazenzlisten gespeichert werden kann, schlägt Agrawal ein Kompressionsverfahren vor, welches mit der Postorder Nummer der Knoten arbeitet, ähnlich des Pre-/Postorder Index für Bäume bei Grust et al. [3].

Jeder Knoten erhält neben seiner Postorder Nummer ein oder mehrere Erreichbarkeitsintervalle. In diesen Intervallen werden die von diesem Knoten aus erreichbaren Postorder Nummern gespeichert in der Form $[von, bis]$. Nun ist genau jeder Knoten, dessen Postorder Nummer in einem der Intervalle enthalten ist, von dem Knoten aus erreichbar, alle anderen nicht. Für eine Erreichbarkeitsanfrage müssen also nur alle Intervalle des Startknotens überprüft werden, ob die Postorder Nummer des Zielknotens darin enthalten ist. Ein Beispiel für diese Indizierung ist in Abbildung 1 dargestellt.

Handelt es sich bei dem Graphen um einen Baum, so hat jeder Knoten nur ein einziges Intervall, in dem alle erreichbaren Knoten enthalten sind; bei DAGs[1] können Knoten

[1] DAG: Directed Acyclic Graph (gerichteter kreisfreier Graph).

mehrere Intervalle benötigen. Handelt es sich bei dem Graphen nicht um einen DAG, sondern um einen Graphen, der auch Zyklen / Kreise enthält, so muss er zunächst in einen DAG überführt werden. Dies geschieht, indem man die starken Zusammenhangskomponenten[2] identifiziert und jede Komponente als einen Knoten ansieht; die Zusammenhangskomponente wird also durch einen neuen Knoten ersetzt. Anschließend werden alle Kanten, die zu einem der in der Zusammenhangskomponente enthaltenen Knoten führten, als eingehende Kante des neuen Knoten hinzugefügt, sowie alle aus einem der Knoten ausgehenden Kanten als ausgehende Kante des neuen Knoten. Der so entstandene Graph besitzt keine Zyklen mehr, ist also ein DAG.

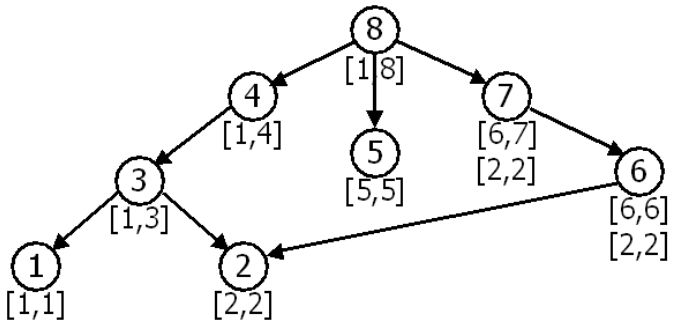

Abbildung 1. Erreichbarkeit im DAG nach Agrawal.

Zur Erstellung des Index wird zunächst ein aufspannender Baum im Graphen bestimmt, und für diesen die Postorder Nummern vergeben. Die Wahl des Baumes ist dabei von großer Wichtigkeit, da der Speicherbedarf der Hülle stark davon abhängt. Agrawal schlägt hier ein Vorgehen vor, mit dem so wenig wie möglich zusätzliche Intervalle im Graphen entstehen. Dabei wird der spannende Baum so erzeugt, dass möglichst lange Wege von der Wurzel zu den Blättern entstehen, anstatt kurzen, weit verzweigten Ästen. Dadurch wird die Anzahl der benötigten zusätzlichen Intervalle minimiert.

In diesem aufspannenden Baum werden nun die Postorder Nummen vergeben, und die initialen Intervalle erstellt. Anschließend werden alle nicht zum Baum gehörenden Kanten betrachtet, und die zusätzlichen Intervalle gespeichert. Damit ist für jeden Knoten gespeichert, welche Knoten von ihm aus erreichbar sind.

Für dieses Speicherverfahren gibt Agrawal auch Update-Strategien für das Hinzufügen und Löschen von Knoten und Kanten an. Auf diese soll in der Studienarbeit nicht weiter eingegangen werden.

[2] Eine starke Zusammenhangskomponente ist ein (größtmöglicher) Teilgraph, in dem jeder Knoten von jedem Knoten des Teilgraphen aus erreichbar ist.

3 Ziele der Arbeit

Im Rahmen dieser Studienarbeit soll die Indizierung von Graphen nach Agrawal implementiert werden. Wichtigster Aspekt dabei ist die Erstellung des Index aus einem gegebenen Graphen, außerdem soll die Beantwortung von Erreichbarkeitsanfragen der Knoten möglich sein. Anschließend soll die Performance der Algorithmen bei Graphen mit unterschiedlichen Eigenschaften untersucht und verglichen werden.

4 Umsetzung

Der Index auf den gegebenen Graphen soll in einer Oracle Datenbank gespeichert werden. Die Implementierung erfolgt über *stored procedures* in der Programmiersprache PL/SQL [1] von Oracle.

Folgende Arbeitsschritte sollen dabei durchgeführt werden:

Ein beliebiger gegebener Graph soll in einen Komponentengraphen überführt werden (dazu wird eine bereits existierende *stored procedure* verwendet). Anschließend wird zu diesem Graphen der aufspannende Baum berechnet; hierbei sollen mehrere Varianten implementiert und verglichen werden: die optimale Erstellung des Baumes nach Agrawal einerseits, sowie ein mit weniger Aufwand erstellter Baum andererseits. Bei letzterem soll für jeden Knoten der Vorgänger nach festen Merkmalen wie Knotenname oder Knotengrad bestimmt werden, sodass der Baum quasi zufällig, aber nachvollziehbar erstellt wird. Dann erfolgt die Erstellung des Index und dessen Speicherung in der Datenbank. Zuletzt soll die Beantwortung von Anfragen an die Datenbank nach Erreichbarkeit der Knoten ermöglicht werden.

Nach erfolgter Implementierung sollen Messungen von Performance und Speicherbedarf mit den verschiedenen Algorithmen sowie verschiedenen Graphen vorgenommen werden. Folgende Punkte sind hierbei von besonderem Interesse:

- Dauer der Erstellung des aufspannenden Baumes je nach verwendetem Algorithmus und Art des Graphen (unterschiedliche Graphgrößen, unterschiedliche Graphdichten).
- Erstellungsdauer und Größe der transitiven Hülle je nach Erstellungsart des Baumes und Art des Graphen.
- Anfragedauer an den Index je nach Erstellungsart des Baumes und Art des Graphen.

Für evtl. auftretetende Unterschiede soll im Rahmen dieser Studienarbeit eine Erklärung gesucht werden. Die Ergebnisse sollen bewertet, und eine Einschätzung der Einsatzfähigkeit des Indexes nach Agrawal vorgenommen werden. Ebenso sollen mögliche Grenzen des Indexes aufgezeit werden.

Literatur

1. Oracle PL/SQL. http://www.oracle.com/technology/tech/pl_sql/.
2. R. Agrawal, A. Borgida, and H. V. Jagadish. Efficient Management of Transitive Relationships in Large Data and Knowledge Bases. In J. Clifford, B. G. Lindsay, and D. Maier, editors, *Proceedings of the ACM SIGMOD International Conference on Management of Data*, pages 253–262, Portland, Oregon, USA, 1989. ACM.
3. T. Grust, M. van Keulen, and J. Teubner. Accelerating XPath Evaluation in any RDBMS. *ACM Trans. Database Syst.*, 29:91–131, 2004.
4. M. Kanehisa, S. Goto, S. Kawashima, Y. Okuno, and M. Hattori. The KEGG resource for deciphering the genome. *Nucleic Acids Res.*, 32(Database issue):D277–D280, 2004.