Baumweite (& Pfadweite)

Exakte Algorithmen für NP-schwere Probleme
Handout vom 21. November 2005

Corinna Habets
exakt@geekin.de

27. November 2005

1 Kontext

Die Baumweite (Pfadweite) sagt aus, wie „baum-ähnlich“ („pfad-ähnlich“) ein gegebener Graph ist. Da viele Algorithmen auf Bäumen (oder Pfaden) effizient laufen, die dies auf allgemeinen Graphen nicht tun, ist es interessant, die Baumweite (Pfadweite) zu kennen.

Das Konzept der Baumweite wird im besprochenen Paper „Algorithms Based on the Treewidth of Sparse Graphs“ vorausgesetzt, deshalb haben wir es ausführlich besprochen. Die Pfadweite ist größtenteils analog zur Baumweite.

2 Cops & Räuber

Als Hinführung auf Baum- & Pfadweite, haben wir das „Cops & Räuber“-Spiel besprochen. (Es lässt sich hinterher übertragen.)

Das Spiel funktioniert so:
In einem ungerichteten Graphen gibt es **einen Räuber** und **mehrere Cops**. Beide Spielfigurtypen halten sich in Knoten auf (immer wenn wir den Graph betrachten).

Der Räuber bewegt sich so:

- Er muss den Kanten des Graphen folgen.
- Er ist beliebig schnell.

Anschaulich kann man sich den Räuber als einen verdammt schnellen Läufer vorstellen, der sich nur auf Strassen (den Kanten des Graphen) bewegen darf.

Die Cops bewegen sich so:

- Sie können von einem Knoten zum anderen springen. (D.h. die Knoten müssen nicht durch eine Kante verbunden sein.
- Sie brauchen aber eine gewisse Zeit dafür.

Anschaulich kann man sich vorstellen, daß jeder Cop in einem Helikopter sitzt und so den Strassenverlauf ignorieren kann.

Wir können die Cops auf Knoten setzen und auch wieder entfernen. Auf den Räuber haben wir keinen Einfluss.

Es gibt zwei Spielvarianten:

1. **Ohne Sehen** - Die Cops sind ahnungslos und haben keinen Schimmer wo der Räuber ist.
2. **Mit Sehen** - Dank flächendeckender Überwachung wissen die Cops genau wo der Räuber ist.

Ziel des Spiels ist es, den Räuber mit der geringstmöglichen Anzahl Cops zu „fangen". Der Räuber ist gefangen, wenn im nächsten Schritt ein Cop auf ihn ziehen wird und der Räuber nicht fliehen kann.[1]

Die geringstmögliche Anzahl Cops um in einem $GraphenG = (V, E)$ den Räuber zu erwischen, bezeichnen wir mit **searcherNumber(G)**. Diese Zahl kann höchstens so groß wie $|V| - 1$ sein.

Es gibt für die Cops eine Menge möglicher Strategien um den Räuber dingfest zu machen. Wie eine gute Strategie aussehen könnte zeigt Abb. 1). (Alle Abbildungen sind auf dem letzten Blatt.) Beim abgebildeten Graph ist der Spielverlauf gleich, egal ob wir mit oder ohne Sehen spielen.

Wenn an jedem Ende einer Kante ein Cop ist, nennen wir diese Kante „gesichert"[2]. In Schritt 2) ist die gesicherte Kante dick eingezeichnet. Wenn wir nun einen der beiden Cops umsetzen würden, könnte der Räuber wieder in diesen Teil des Graphen kommen. Wir wollen aber, daß einmal gesicherte Kanten auch sicher, d.h. räuberfrei, bleiben. Deshalb brauchen wir einen dritten Cop. Wenn er den benachbarten Knoten besetzt, können wir den mittleren Cop gefahrlos umsetzen. Die gestrichelte Kante ist also immer noch gesichert. Nach diesem Schema können wir auch den restlichen Graphen sichern. Daher ist searcherNumber(G in Abb.1) = 3.

[1] Vielleicht fragst Du Dich jetzt, wo der Witz bei der Variante „Mit Sehen" ist. Kann man denn nicht immer mit einem Cop auskommen, wenn man weiss in welchem Knoten der Räuber ist? Indem man einfach den Cop auf diesen Knoten setzen, fertig? Funktioniert leider nicht. Um das zu sehen, ist die Läufer/Helikopter-Analogie sehr hilfreich: Der einzelne Cop landet mit seinem Hubschrauber also auf dem Platz (Knoten) des Räubers. Der Räuber ist aber beliebig schnell und verdrückt sich in einen anderen Teil der Stadt (des Graphen). Wenn der Cop gelandet ist, also zu dem Zeitpunkt an dem wir die Situation betrachten, ist der Räuber längst woanders.

[2] Englisch: checked

3 Baum- & Pfadweite

3.1 Definitionen

Jetzt haben wir genug vorbereitet, um das eigentliche Thema anzugehen. Zunächst ein paar wichtige **Definitionen**

Die **Baumweite** eines Graphen G ist definiert als die kleinste **Weite** aller **Baumzerlegungen** von G.

Eine **Baumzerlegung** von $G = (V, E)$ ist ein Paar (X, T), wobei $T = (I, F)$ ein Baum ist und $X = \{X_i | i \in I\}$ eine Familie von Untermengen von V, eine für jeden Knoten in V, so daß gilt:

- $\bigcup_{i \in I} X_i = V$
- für alle Kanten $(v, w) \in E$ gibt es ein $i \in I$ mit $v, w \in X_i$
- für alle $i, j, k \in I$ gilt, wenn j auf dem Pfad von i zu k in T ist, dann $X_i \cap X_k \subseteq X_j$

Verständlicher ausgedrückt, verteilen wir die Knoten von G in Taschen[3], die in einer Baumstruktur angeordnet sind. Siehe dazu auch Abb. 2!
Dabei gelten folgende Regeln:

- jeder Knoten aus V ist in mindestens einer Tasche
- die beiden Endknoten jeder Kante sind zusammen in mindestens einer Tasche
- für jeden Knoten v bilden die Taschen, die v enthalten, einen Unterbaum

Die **Weite** w() einer Baumzerlegung ist die maximale Anzahl Knoten in einer einzelnen Tasche - 1.

Das obige wird zur Definition von **Pfadweite** indem man überall „Baum“ durch „Pfad“ ersetzt.

3.2 Rückbezug auf „Cops & Räuber“

Leider gibt es sehr viele verschiedene mögliche Baumzerlegungen. Für die Baumweite brauchen wir jedoch die Zerlegung mit der kleinsten Weite. Dieses Problem ist NP-schwer. Auch der Algorithmus ist kompliziert. Deshalb beschränken wir uns auf eine intuitive Darstellung und kommen dazu auf das „Cops & Räuber“-Spiel zurück.

Praktischerweise entspricht die minimale Anzahl benötigter Cops genau der Anzahl Knoten in der größten Tasche der minimalen Zerlegung.

[3] Englisch: buckets oder bags

- Für „C & R“ mit Sehen, gilt $searcherNumber(G) - 1 = Baumweite(G)$
- Für „C & R“ ohne Sehen, gilt $seacherNumber(G) - 1 = Pfadweite(G)$

Wenn wir „C & R“mit Sehen spielen, repräsentiert die Baumzerlegung die Fallunterscheidung beim Sichern. Entweder der Räuber ist rechts oder links ...

Wenn wir ohne Sehen spielen repräsentiert die Pfadzerlegung eine Art Sweep-Line, mit der wir den ganzen Graphen deterministisch, linear durchgehen und sichern.

Betrachten wir zur Verdeutlichung den Binärbaum B in Abb. 3). Dort ist die

- $Baumweite(B) = 1$[4]
- $Pfadweite(B) = Höhe(B) \approx \log |V|$[5]

4 Fazit

Sowohl die minimale Baum- als auch Pfadzerlegung sind NP-schwer. Trotzdem sind Baum- und Pfadweite interessante Kennzahlen eines Graphen. Wie eingangs erwähnt gibt es viele Algorithmen die nur auf Bäumen oder Pfaden effizient laufen. Diese Algorithmen laufen aber unter Umständen schon besser auf Baum- oder Pfadzerlegungen beliebiger Graphen. Insgesamt also ein spannendes Thema.

5 Abbildungen

LaTex und Bilder/Graphen sind bei mir ein Problem. Leider hatte ich nicht genug Zeit es für dieses Handout zu lernen. Ich entschuldige mich für das Hin- und Herblättern.

[4] Die Baumweite von Bäumen ist immer 1 = größtmögliche Baumähnlichkeit.

[5] Die Pfadweite ist immer größer oder gleich der Baumweite, da jeder Pfad auch ein Baum ist.