Das algebraische Pfadproblem

Eric Kunze

5. Januar 2022

Mit dieser Lösung ist keine Garantie auf Vollständigkeit und/oder Korrektheit verbunden!

1 Verallgemeinerung und Semiringe

Wir kennen bereits den Floyd-Warshall-Algorithmus zur Berechnung kürzester Wege. Grundlage des dynamischen Verfahrens war die schrittweise Zulassung weiterer Knoten, bis am Ende schließlich alle Knoten des Graphen zugelassen sind und die entstandene Matrix die tatsächliche, gesuchte Distanzmatrix ist. Bei der Zulassung des nächsten Knotens werden die Matrixeinträge mithilfe der Update-Formel

$$D_G^{(k+1)}(u,v) = \min\left\{D_G^{(k)}(u,v), D_G^{(k)}(u,k+1) + D_G^{(k)}(k+1,v)\right\}$$

aktualisiert. In dieser Formel sind im Wesentlichen zwei Operationen beteiligt: die Addition, die die Länge des neuen Alternativweges $u \rightsquigarrow k+1 \rightsquigarrow v$ berechnet, und das Minimum, mit welchem wir die zwei alternativen Wege beurteilen.

Man kann nun diese beiden Operationen geeignet austauschen und andere Problemstellungen lösen. Dazu betrachten wir eine allgemeine algebraische Strukur: Semiringe.

Definition 1.1 (Semiring)**.** *Ein* ***Semiring*** *ist ein Tupel* $(S, \oplus, \odot, \mathbf{0}, \mathbf{1})$ *bestehend aus*

- *einer Trägermenge* S*,*
- *einer binären Operation* $\oplus$ *(Akkumulationsoperation) mit neutralem Element* $\mathbf{0}$ *und*
- *einer binären Operation* $\odot$ *(Pfadoperation) mit neuralem Element* $\mathbf{1}$*.*

Weiterhin fordern wir folgende Eigenschaften:

- $\oplus$ *ist assoziativ und kommutativ,*
- $\odot$ *ist assoziativ,*

- $\odot$ *ist distributiv über* $\oplus$*, d.h.* $a \odot (b \oplus c) = (a \odot b) \oplus (a \odot c)$*, und*
- $\mathbf{0}$ *ist ein Annihilator für* $\odot$*, d.h.* $a \odot \mathbf{0} = \mathbf{0}$*.*

Ein Semiring heißt ***idempotent****, falls* $s \oplus s = s$ *für alle* $s \in S$ *gilt.*

Die eigentliche Definition eines Semirings ist für uns nicht weiter relevant, sie sichert lediglich, dass alles so funktioniert, wie wir es uns wünschen. Viel bedeutender sind die konkreten Problemstellungen.

Beispiel 1.2 (Kürzeste-Wege-Problem)**.** *Für das Kürzeste-Wege-Problem haben wir uns die Bedeutung der Operationen oben schon überlegt. Wir wählen also den sogenannten tropischen Semiring*

$$(S, \oplus, \odot, \mathbf{0}, \mathbf{1}) = (\mathbb{R}^{\infty}_{\geq 0}, \min, +, \infty, 0) \ . \qquad \text{(tropischer Semiring)}$$

Beispiel 1.3 (Kapazitätsproblem)**.** *Die Kanten unserer Graphen seien (Einbahn-)Straßen mit maximalen Belastungsgewichten. Nun ist die Frage, wieviel ein Lkw maximal wiegen darf, um zwischen zwei Knoten verkehren verkehren zu dürfen? Entlang eines Pfades ist die maximale Belastung durch den Abschnitt bestimmt, der am wenigsten Gewicht trägt – also die minimale Belastung einer Kante. Von zwei Pfaden wählen wir denjenigen, der mehr Belastung zulässt – also das Maximum der beiden Pfade. Nehmen wir nun noch an, dass alle Gewichte natürliche Zahlen oder* ∞ *sind, dann erhalten wir den Semiring*

$$(S, \oplus, \odot, \mathbf{0}, \mathbf{1}) = (\mathbb{N}_{\infty}, \max, \min, 0, \infty) \ . \qquad \text{(Kapzitätsproblem)}$$

Beispiel 1.4 (Zuverlässigkeitsproblem)**.** *Die Gewichte an den Kanten sind Wahrscheinlichkeiten* $p \in [0, 1]$*, wie zuverlässig ein Fahrzeug zwischen zwei Knoten verkehrt. Entlang eines Pfades multiplizieren wir die Wahrscheinlichkeiten um die Gesamtwahrscheinlichkeit vom Start des Pfades zum Ende des Pfades zu gelangen. Von zwei Pfaden wählen wir erneut denjenigen mit größerer Gesamtwahrscheinlichkeit. Der sogenannte Viterbi-Semiring ist damit gegeben durch*

$$(S, \oplus, \odot, \mathbf{0}, \mathbf{1}) = ([0, 1], \max, \cdot, 0, 1) \ . \qquad \text{(Viterbi-Semiring)}$$

Ein weiteres, einfacheres Problem ist das **Erreichbarkeitsproblem** mit $(S, \oplus, \odot, \mathbf{0}, \mathbf{1}) = ([0, 1], \max, \cdot, 0, 1)$. Das letzte für uns relevante, jedoch deutlich komplizierte Problem, für das die Ausführungen hier nicht mehr alle richtig sind, ist das **Prozessproblem**, welches mit formalen Sprachen arbeitet.

Die Verallgemeinerung des Kürzesten-Wege-Problems auf allgemeine Semiringe wird **al-**

gebraisches Pfadproblem genannt. Der zugehörige Algorithmus heißt Aho-Hopcraft-Ullman-Algorithmus. Dieser arbeit jedoch sehr ähnlich zum Floyd-Warshall-Algorithmus.

2 Der Aho-Hopcraft-Ullman-Algorithmus

Betrachten wir einen allgemeinen Semiring $(S, \oplus, \odot, \mathbf{0}, \mathbf{1})$ mit einer Trägermenge S, einer Pfadoperation $\odot$ mit neutralem Element $\mathbf{1}$ sowie einer Akkumulationsoperation $\oplus$ mit neutralem Element $\mathbf{0}$.

Auch der Aho-Hopcraft-Ullman-Algorithmus arbeitet mit Approximationen $D_G^{(k)}$ durch schrittweise Zulassung von mehr inneren Knoten. Wir starten erneut mit der modifizierten Adjazenzmatrix, jedoch ist die Modifikation nun ein wenig anders. Für *verschiedene* Knoten $u, v \in V$, für die eine Kante von u nach v existiert, nutzen wir dieses Kantengewicht (wie in jeder Adjazenzmatrix). Existiert keine Kante von u nach v, so nutzen wir nicht mehr unbedingt ∞, weil es nicht notwendigerweise um echte Entfernungen geht. Im Beispiel des Kapazitätsproblems müssen wir uns fragen, wieviel Gewicht über eine Straße fahren darf, damit diese Straße nicht benutzbar ist – hier ist die Antwort 0. Wir verwenden also im allgemeinen Semiring stets $\mathbf{0}$, das neutrale Element von $\oplus$. Für einen Knoten $v \in V$ können wir immer durch "Stehenbleiben" den besten Weg finden und setzen daher diesen Wert daher auf $\mathbf{1}$. Zusammengefasst ist

$$mA_G = \begin{cases} A_G(u,v) & \text{wenn } u \neq v \\ A_G(u,v) \oplus \mathbf{1} & \text{wenn } u = v \end{cases}$$

und $D_G^{(0)} = mA_G$.

Die Update-Formel im Aho-Hopcraft-Ullman-Algorithmus sieht wie folgt aus:

$$D_G^{(k+1)}(u,v) = D_G^{(k)}(u,v) \oplus \left(D_G^{(k)}(u,k+1) \odot (D_G^{(k)}(k+1,k+1))^* \odot D_G^{(k)}(k+1,v) \right) \quad .$$

Im Vergleich zum Floyd-Warshall-Algorithmus enthält sie damit einen "Faktor" mehr – nämlich $(D_G^{(k)}(k+1,k+1))^*$.

Dieser ist jedoch für viele praktische Problemstellungen, die wir betrachten, zu vernachlässigen.Wir werden in Abschnitt 4 begründen, wieso das der Fall ist.

3 Ein bisschen Abstraktes

Dafür benötigen wir zunächst einmal ein tieferes Verständnis über die Verallgemeinerungen, die wir gemacht haben, wenn wir über Semiringe sprechen.

Manchmal benötigt man viele Operationen hintereinander. Für die gewöhnliche Addition kennt man daher das Summenzeichen $\sum$, um z.B. $\sum_{i=0}^{n} i = 0+1+2+\cdots+n$ abzukürzen. Man kann auch die Indexmenge, die i durchläuft unten an das Summenzeichen schreiben, d.h.

$$\sum_{i=0}^{n} i = \sum_{i\in\{0,\ldots,n\}} i.$$

Diese Prozedur kann man auch für andere Operationen machen, z.B. für

$$s_0 \oplus s_1 \oplus s_2 \oplus s_3 \oplus \cdots \oplus s_n$$

wäre eine abkürzende Schreibweise wünschenswert. Wir nutzen wieder das Summenzeichen und müssen aber die Operation, die verallgemeinert werden soll (hier: $\oplus$), mit notieren. Daraus ergibt sich

$$\sum_{i\in\{0,\ldots,n\}}^{\oplus} s_i := s_0 \oplus s_1 \oplus s_2 \oplus s_3 \oplus \cdots \oplus s_n \ .$$

Beispiel 3.1. *Für das Kürzeste-Wege-Problem ist* $\oplus = \min$ *und daher*

$$\sum_{i\in\{5,6,\ldots,10\}}^{\oplus} i = \sum_{i\in\{5,6,\ldots,10\}}^{\min} i = \min\{5,6,7,8,8,10\} = 5 \ .$$

Für das Kapazitätsproblem dagegen ist $\oplus = \max$ *und somit*

$$\sum_{i\in\{5,6,\ldots,10\}}^{\oplus} i = \sum_{i\in\{5,6,\ldots,10\}}^{\max} i = \max\{5,6,7,8,8,10\} = 10 \ .$$

Formal kann man $\sum^{\oplus}$ als Abbildung auffassen, die *vielen* Elementen ein Ergebnis zuordnet, während $\oplus$ immer *genau zwei* Elementen ein Ergebnis zuordnet. Stellt man an $\sum^{\oplus}$ nun noch weitere Anforderungen, auf die wir hier nicht näher eingehen wollen, und nennt sie dann vollständig, dann kann man das algebraische Pfadproblem genau definieren.

Definition 3.2. *Das algebraische Pfadproblem auf einem* $\sum^{\oplus}$*-vollständigen, idempotenten Semiring* $(S, \oplus, \odot, \mathbf{0}, \mathbf{1})$ *besteht in der Berechnung von*

$$D_G(u,v) = \sum_{p\in P_{u,v}}^{\oplus} c(p)$$

Hinweis 3.3. *Manche Autoren verwenden statt* $\sum^{\oplus}$ *auch die Notation eines großen* $\oplus$*, also*

$$s_0 \oplus s_1 \oplus \cdots \oplus s_n = \sum_{i\in\{0,1,\ldots,n\}}^{\oplus} s_i = \bigoplus_{i\in\{0,\ldots,n\}} s_i = \bigoplus_{i=0}^{n} s_i.$$

Für ein Element $s \in S$ in einem Semiring $(S, \oplus, \odot, \mathbf{0}, \mathbf{1})$ können wir allgemein eine "Potenz" definieren.

Definition 3.4. *Für ein $s \in S$ ist s^n induktiv definiert durch*

$$s^0 := \mathbf{1} \qquad \textit{und} \qquad s^{n+1} := s \odot s^n \quad \textit{für alle } n \in \mathbb{N} \ .$$

Zum Schluss kann man noch so etwas wie einen "Abschluss der Potenz" definieren, den **Stern**. Die Definition ist analog zu der des Kleene-Sterns für Formale Sprachen.

Definition 3.5. *Für ein $s \in S$ ist der **Stern** s^* definiert als*

$$s^* := \sum_{n \in \mathbb{N}}^{\oplus} s^n \ .$$

Hinweis 3.6. *Ist S die Menge aller Sprachen, $s \in S$ also eine Sprache, $\odot$ die Konkatenation von Wörtern und $\oplus = \cup$ die Vereinigung von Mengen bzw. Sprachen, so ergibt sich*

$$s^* = \sum_{n \in \mathbb{N}}^{\cup} s^n = \bigcup_{n \in \mathbb{N}} s^n \ .$$

Ersetzt man s durch L so erkennt man genau die Definition des Kleene-Sterns für formale Sprachen.

4 Eine wesentliche Vereinfachung

Die Update-Formel des Aho-Hopcraft-Ullman-Algorithmus enthält den zusätzlichen "Faktor" $D_G^{(k)}(k+1, k+1)^*$. Bei allen unseren praktischen Problemen – *außer dem Prozessproblem* – fällt dieser Teil nämlich weg. Wieso dies so ist, wollen wir hier begründen, indem wir zeigen, dass stets $s^* = \mathbf{1}$ für alle $s \in S$.

Wir vollführen das Ganze anhand des Kapazitätsproblems aus Beispiel 1.3 und fixieren daher den Semiring $(S, \oplus, \odot, \mathbf{0}, \mathbf{1}) = (\mathbb{N}_\infty, \max, \min, 0, \infty)$.

Gemäß Definition 3.5 gilt $s^* = \sum_{n \in \mathbb{N}}^{\oplus} s^n$ mit $s^0 = \mathbf{1}$ und $s^{n+1} = s \odot s^n$. Im unserem Semiring gilt nach Definition 3.4 für die Potenzen:

- $s^0 = \mathbf{1} = \infty$
- $s^1 = s \odot s^0 = \min\{s, \infty\} = s$
- $s^2 = s \odot s^1 = \min\{s, s\} = s$
- ...

Man kann nun mittels vollständiger Induktion noch zeigen, dass stets $s^n = s$ für alle $n \geq 1$.

Schließlich ist

$$s^* = \sum_{n\in\mathbb{N}}^{\oplus} s^n = \sum_{n\in\mathbb{N}}^{\max} s^n = \sup\{s^n : n \in \mathbb{N}\} = \sup\{\infty, s, s, \ldots\} = \infty = \mathbf{1} \ .$$

Somit gilt dann in der Updateformel

$$\begin{aligned} D_G^{(k+1)}(u,v) &= \max\left\{D_G^{(k)}(u,v), \min\left\{D_G^{(k)}(u,k+1), \infty, D_G^{(k)}(k+1,v)\right\}\right\} \\ &= \max\left\{D_G^{(k)}(u,v), \min\left\{D_G^{(k)}(u,k+1), \quad D_G^{(k)}(k+1,v)\right\}\right\} \\ &= D_G^{(k)}(u,v) \oplus \left(D_G^{(k)}(u,k+1) \odot D_G^{(k)}(k+1,v)\right) \ . \end{aligned}$$

So wie wir diese Eigenschaft hier für das Kapazitätsproblem gezeigt haben, kann man für alle weiteren Probleme (außer dem Prozessproblem) verfahren. Insbesondere kann man damit auch zeigen, dass der Floyd-Warshall-Algorithmus für kürzeste Wege wirklich ein Spezialfall des Aho-Hopcraft-Ullman-Algorithmus ist.