5 Zwei spieltheoretische Aspekte

In diesem Kapitel wollen wir uns mit dem algorithmischen Problem beschäftigen, sogenannte Und-Oder-Bäume (kurz UOB) auszuwerten. Sie sind ein Spezialfall von Spielbäumen, der selbst aber auch eine Rolle spielt, zum Beispiel in manchen Theorembeweisern. Wir beschränken uns der Bequemlichkeit halber im folgenden auf Bäume mit Verzweigungsgrad 2. Man kann aber analoge Ergebnisse allgemein für Verzweigungsgrad $d \geqslant 2$ beweisen.

5.1 Und-Oder-Bäume und ihre deterministische Auswertung

5.1 Definition Für $k \geqslant 1$ sei T_k der wie folgt rekursiv definierte vollständige binäre Baum der Höhe $2k$, dessen innere Knoten abwechselnd mit "$\wedge$" und "$\vee$" markiert sind.

- Die Wurzel von T_1 ist ein $\wedge$-Knoten und hat zwei $\vee$-Knoten als Nachfolger. Jeder dieser Knoten hat zwei Blätter als Nachfolger.
- Für $k \geqslant 2$ ergibt sich T_k aus T_1, indem man dessen Blätter durch Kopien von T_{k-1} ersetzt. $\diamond$

Wie man leicht sieht, könnte man in obiger Definition den Rekursionschritt auch völlig äquivalent so formulieren:

- Für $k \geqslant 2$ und $1 \leqslant l < k$ ergibt sich T_k aus T_l, indem man dessen Blätter durch Kopien von T_{k-l} ersetzt.

Im folgenden bezeichne n stets die Anzahl der Blätter eines UOB. T_k besitzt also $n = 4^k$ Blätter, die im folgenden mit $x_1, \ldots, x_{4^k}$ bezeichnet werden.

5.2 Durch die Festlegung von booleschen Werten an allen Blättern eines UOB wird auf naheliegende Weise auch für alle inneren Knoten und damit auch für die Wurzel des Baumes ein Wert festgelegt.

In den beiden ersten Abschnitten dieses Kapitels wollen wir uns mit deterministischen und einem randomisierten Algorithmus zur Bestimmung der Wurzelwerte von UOB beschäftigen. Dabei wollen wir uns insbesondere dafür interessieren, wieviele Blätter der Algorithmus besucht, um den Wurzelwert zu bestimmen.

5.3 Offensichtlich kann durch den Besuch aller $n = 4^k$ Blätter und die Berechnung der Werte aller inneren Knoten „bottom up" den der Wurzel bestimmen.

Zunächst stellt sich die Frage, ob deterministische Algorithmen auch geschickter vorgehen können. Die Antwort ist nein:

5.4 Satz. *Für jedes $k \geqslant 1$ und jeden deterministischen Algorithmus A zur Auswertung von UOB gilt: Es gibt eine Folge $x_1, \ldots, x_{4^k}$ von Bits, so dass A bei der Auswertung von T_k mit den x_i als Blattwerten* alle $n = 4^k$ *Blätter besucht. Dabei ist der Wert der Wurzel gleich dem des zuletzt besuchten Blattes und es kann also sowohl erzwungen werden, dass dieser gleich 0 ist, als auch, dass er gleich 1 ist.*

5.5 Beweis. Durch Induktion:

$k = 1$**:** Es ist klar, dass A mindestens ein Blatt besuchen muss. O. B. d. A. sei dies x_1. Wir setzen $x_1 = 0$. Damit ist weder der Wert des übergeordneten $\vee$-Knotens noch der der Wurzel bereits festgelegt und A muss ein weiteres Blatt besuchen. Wieder gibt es o. B. d. A. zwei Möglichkeiten:

1. Das als zweites besuchte Blatt ist x_2. Wir setzen $x_2 = 1$. Damit ist nur der Wert des übergeordneten $\vee$-Knotens klar aber noch nicht der der Wurzel und A muss ein weiteres Blatt besuchen. O. B. d. A. sei dies x_3. Wir setzen $x_3 = 0$. Damit muss A auch noch x_4 besuchen, denn dessen Wert ist der der Wurzel.
2. Das als zweites besuchte Blatt ist x_3. Wir setzen $x_3 = 0$. Damit ist weder der Wert des übergeordneten $\vee$-Knotens noch der der Wurzel bereits festgelegt und A muss ein weiteres Blatt besuchen. O. B. d. A. sei dies x_2. Wir setzen $x_2 = 1$. Damit muss A auch noch x_4 besuchen, denn dessen Wert ist der der Wurzel.

$k - 1 \rightsquigarrow k$**:** Wir fassen T_k auf als einen T_1-Baum, dessen Blätter durch T_{k-1}-Bäume ersetzt sind. Wir bezeichnen die „Blätter" von T_1 mit $y_1, \ldots, y_4$. Analog zur überlegung für den Induktionsanfang ist klar, dass A mindestens einen der Werte y_i bestimmen muss. Mehr noch, man kann durch geschickte Wahl der y_i in Abhängigkeit von der Reihenfolge, in der A sie berechnet, erzwingen, dass A sogar *alle* Werte y_1, y_2, y_3 und y_4 ermitteln muss. Nach Induktionsvoraussetzung gibt es für jeden der T_{k-1}-Bäume eine Belegung der Blattwerte, die das gewünschte y_i liefert und gleichzeitig erzwingt, dass A zu dessen Berechnung jeweils alle darunter liegenden Blätter besuchen muss.

Also muss A in diesem Fall alle Blätter überhaupt besuchen.

■

5.2 Analyse eines randomisierten Algorithmus für die Auswertung von UOB

5.6 Algorithmus.

```
proc AndNodeEval(T)
if IsLeaf(T) then return value(T) fi
⟨andernfalls:⟩
T' ← ⟨zufällig gewählter Unterbaum von T⟩
r ← OrNodeEval(T')
if r = 0 then
    return 0
else
    T'' ← ⟨der andere Unterbaum von T⟩
    return OrNodeEval(T'')
fi

proc OrNodeEval(T)
```



```
T' ← ⟨zufällig gewählter Unterbaum von T⟩
r ← AndNodeEval(T')
if r = 1 then
    return 1
else
    T'' ← ⟨der andere Unterbaum von T⟩
    return AndNodeEval(T'')
fi
AndNodeEval(root)
```

5.7 Satz. *Der Erwartungswert für die Anzahl der von Algorithmus 5.6 besuchten Blätter für einen T_k-Baum ist für jede Folge $x_1, \ldots, x_{4^k}$ von Blattwerten höchstens $3^k = n^{\log_4 3} \approx n^{0.792\ldots}$.*

5.8 Beweis. Durch Induktion.

$k = 1$**:** Diesen Fall erledigt man durch systematisches überprüfen aller 16 möglichen Kombinationen für die $x_1, \ldots, x_4$. Beispielhaft betrachten wir den Fall 0100:

1. Falls zuerst der linke Teilbaum ausgewertet wird: Mit gleicher Wahrscheinlichkeit wird erst und nur die 1 oder erst die 0 und danach die 1 besucht. Anschließend werden im rechten Teilbaum beide Blätter besucht. Erwartungswert: $7/2$.
2. Falls zuerst der rechte Teilbaum untersucht wird: Nach dem Besuch beider Blätter ist klar, dass der T_1-Baum den Wert 0 liefert. Erwartungswert: 2.

Da beide Fälle gleich wahrscheinlich sind, ergibt sich insgesamt $1/2 \cdot 7/2 + 1/2 \cdot 2 = 11/4 < 3$.

$k-1 \rightsquigarrow k$**:** Wir betrachten zunächst nicht einen ganzen T_k-Baum, sondern einen $\vee$-Knoten, an dem zwei T_{k-1}-Bäume „hängen“. Es gibt zwei Fälle:

O1. Der $\vee$-Knoten wird eine 1 liefern: Dann muss mindestens einer der T_{k-1}-Bäume dies auch tun. Da gleichwahrscheinlich jeder der beiden zuerst untersucht wird, wird mit einer Wahrscheinlichkeit $p \geqslant 1/2$ als erstes ein (und nur ein) Unterbaum untersucht, der eine 1 liefert. Mit Wahrscheinlichkeit $1-p \leqslant 1/2$ werden beide Unterbäume untersucht. Der Erwartungswert ist also höchstens $p \cdot 3^{k-1} + (1-p) \cdot 2 \cdot 3^{k-1} = (2-p) \cdot 3^{k-1} \leqslant 3/2 \cdot 3^{k-1}$.

O2. Der $\vee$-Knoten wird eine 0 liefern: Dann müssen beide T_{k-1}-Bäume dies auch tun. Mit der Induktionsvoraussetzung ergibt sich, dass der Erwartungswert für die Anzahl der in diesem Fall besuchten Blätter höchstens $2 \cdot 3^{k-1}$ ist.

Betrachten wir nun die Wurzel des T_k-Baumes, an der zwei der eben untersuchten Bäume hängen. Es gibt zwei Fälle:

U1. Der $\wedge$-Knoten wird eine 0 liefern: Dann muss mindestens einer der Unterbäume dies auch tun. Da gleichwahrscheinlich jeder der beiden zuerst untersucht wird, wird mit einer Wahrscheinlichkeit $p \geqslant 1/2$ als erstes ein (und nur ein) Unterbaum untersucht, der eine 0 liefert. Mit Wahrscheinlichkeit $1-p \leqslant 1/2$ werden beide Unterbäume untersucht. Gemäß der überlegungen in O1. und O2. ist der Erwartungswert folglich höchstens $p \cdot 2 \cdot 3^{k-1} + (1-p) \cdot (3/2 \cdot 3^{k-1} + 2 \cdot 3^{k-1}) = 7/2 \cdot 3^{k-1} - p \cdot 3/2 \cdot 3^{k-1} \leqslant 11/4 \cdot 3^{k-1} \leqslant 3^k$.

U2. Der $\wedge$-Knoten wird eine 1 liefern: Dann müssen beide Unterbäume dies auch tun. Nach Fall O1. ist daher der Erwartungswert für die Anzahl besuchter Blätter $2 \cdot 3/2 \cdot 3^{k-1} \leqslant 3^k$.

■

5.3 Zwei-Personen-Nullsummen-Spiele

In diesem Abschnitt sind ein wenig Notation und Ergebnisse zu einem Thema aus der Spieltheorie zusammen gestellt.

5.9 Im allgemeinen hat man es mit $n \geqslant 2$ *Spielern* zu tun. Jeder Spieler i hat eine (endliche) Menge S_i sogenannter *reiner Strategien* s_j^i zur Auswahl. Für jeden Spieler i gibt es eine Funktion $u_i : S_1 \times \cdots \times S_n \to \mathbb{R}$, die für jede Kombination von Strategien angibt, welchen *Nutzen* oder *Gewinn* Spieler i hat, wenn die Spieler sich für eine bestimmte Kombination von Strategien entscheiden.

5.10 Bei *Zwei-Personen-Nullsummen-Spielen* gibt es $n = 2$ Spieler und für die Nutzenfunktionen gilt: $u_1 = -u_2$. Es genügt also zum Beispiel u_1 anzugeben; das kann man dann in Form einer Matrix $\mathbf{M}$ mit $|S_1|$ Zeilen und $|S_2|$ Spalten tun, bei der Eintrag M_{ij} gerade $u_1(s_i^1, s_j^2)$ ist.

Deshalb spricht man dann auch manchmal vom *Zeilenspieler* und vom *Spaltenspieler*. Identifiziert man die Wahl einer reinen Strategie i mit dem Einheitsvektor $\mathbf{e}_i$ (jeweils passender Länge und in Spaltenform), dann ist $u_1(s_i^1, s_j^2) = \mathbf{e}_i^T \mathbf{M} \mathbf{e}_j$.

5.11 Eine *gemischte Strategie* ist eine Wahrscheinlichkeitsverteilung $\mathbf{p}$ auf der Menge der reinen Strategien eines Spielers.

Sind $\mathbf{p}$ und $\mathbf{q}$ gemischte Strategien für Zeilen- und Spaltenspieler, dann ist $\mathbf{p}^T \mathbf{M} \mathbf{q}$ der zu erwartende Gewinn für den Zeilenspieler.

5.12 Satz. (Neumann 1928) *Für Zwei-Personen-Nullsummen-Spiele mit Matrix* $\mathbf{M}$ *gilt:*

$$\max_{\mathbf{p}} \min_{\mathbf{q}} \mathbf{p}^T \mathbf{M} \mathbf{q} = \min_{\mathbf{q}} \max_{\mathbf{p}} \mathbf{p}^T \mathbf{M} \mathbf{q}$$

Wir werden diesen Satz hier nicht beweisen. Man kennt verschiedene Möglichkeiten, es zu tun. Zum Beispiel kann man Verteilungen $\mathbf{p}^*$ und $\mathbf{q}^*$, für die der Wert aus von Neumanns Satz angenommen wird, nach Brouwers Fixpunktsatz als Fixpunkt einer geeigneten Abbildung erhalten.

5.13 Korollar. (Loomis 1946) Für Zwei-Personen-Nullsummen-Spiele mit Matrix $\mathbf{M}$ gilt:

$$\max_{\mathbf{p}} \min_{j} \mathbf{p}^T \mathbf{M} \mathbf{e}_j = \min_{\mathbf{q}} \max_{i} \mathbf{e}_i^T \mathbf{M} \mathbf{q}$$

5.14 Beweis. Es genügt zu zeigen, dass für jedes $\mathbf{p}$ gilt: $\min_{\mathbf{q}} \mathbf{p}^T \mathbf{M} \mathbf{q} = \min_j \mathbf{p}^T \mathbf{M} \mathbf{e}_j$ und analog für die rechten Seiten der beiden Gleichungen aus Satz 5.12 und Korollar 5.13.

Für beliebiges $\mathbf{p}$ ist $\mathbf{p}^T \mathbf{M}$ ein Zeilenvektor $\mathbf{v}^T$. Es sei j eine Stelle in $\mathbf{v}$, an der der kleinste aller in $\mathbf{v}$ vorkommenden Werte steht. Dann ist offensichtlich $\mathbf{v}^T \mathbf{e}_j$ der kleinste überhaupt mögliche Wert, der für ein $\mathbf{v}^T \mathbf{q}$ auftreten kann. ■

Aus Korollar 5.13 ergibt sich offensichtlich die folgende Aussage, die wir im anschließenden Abschnitt ausnutzen werden.

5.15 Korollar. Für alle Verteilungen $\mathbf{p}$ und $\mathbf{q}$ gilt:

$$\min_{j} \mathbf{p}^T \mathbf{M} \mathbf{e}_j \leqslant \max_{i} \mathbf{e}_i^T \mathbf{M} \mathbf{q}$$

5.4 Untere Schranken für randomisierte Algorithmen

Im letzten Abschnitt dieses Kapitels soll eine Technik vorgestellt werden, um untere Schranken für den Ressourcenverbrauch randomisierter Algorithmen nachzuweisen. Tatsächlich handelt es sich wohl um die derzeit einzige solche Methode.

5.16 Stellen Sie sich nun vor, dass es zwei Spieler gibt:

- Spaltenspieler ist jemand der als verschiedene Strategien deterministische Algorithmen A zur Auswahl hat.
- Zeilenspieler ist ein böser Widersacher, der als verschiedene Strategien Eingaben I zur Auswahl hat.

Der Gewinn des Widersachers ist jeweils $C(I, A)$. Das sei zum Beispiel die Laufzeit von Algorithmus A für Eingabe I (oder der Verbrauch irgendeiner anderen Ressource).

Der Widersacher versucht, $C(I, A)$ zu maximieren, der Algorithmenentwerfer versucht, $C(I, A)$ zu minimieren.

- Eine gemischte Strategie des Widersachers ist eine Wahrscheinlichkeitsverteilung auf der Menge der Eingaben.
- Eine gemischte Strategie des Algorithmenentwerfers ist ein randomisierter Algorithmus.

Stellt man sich nun noch vor, dass $\mathbf{M}$ die Werte $C(I, A)$ enthält, dann ist klar:

5.17 Satz. (Minimax-Methode von Yao) *Es sei P ein Problem für eine endliche Menge $\mathcal{I}$ von Eingaben gleicher Größe n und $\mathcal{A}$ eine endliche Menge von Algorithmen für dieses Problem. Für $I \in \mathcal{I}$ und $A \in \mathcal{A}$ bezeichne $C(I, A)$ den Ressourcenverbrauch, z. B. die Laufzeit, von Algorithmus A für Eingabe I.*

Weiter bezeichne $\mathbf{p}$ bzw. $\mathbf{q}$ eine Wahrscheinlichkeitsverteilung auf $\mathcal{I}$ bzw. $\mathcal{A}$. Mit I_p bzw. A_q werde ein gemäß der Verteilung $\mathbf{p}$ bzw. $\mathbf{q}$ aus $\mathcal{I}$ bzw. $\mathcal{A}$ gewählte Eingabe bzw. Algorithmus bezeichnet.

Dann gilt für alle $\mathbf{p}$ und $\mathbf{q}$:

$$\min_{A \in \mathcal{A}} \mathbf{E}\left[C(I_p, A)\right] \leqslant \max_{I \in \mathcal{I}} \mathbf{E}\left[C(I, A_q)\right]$$

5.18 Einige Erläuterungen erscheinen angebracht:

- Der Erwartungswert auf der linken Seite ergibt sich durch die zufällige Wahl von I_p gemäß Verteilung $\mathbf{p}$. Für jeden deterministischen Algorithmus A handelt es sich dabei also um die „erwartete Laufzeit“ von A für gewisse Eingabeverteilungen. Das Minimum der Erwartungswerte, also der Erwartungswert für den „besten“ Algorithmus ist in der Ungleichung von Bedeutung.
- Der Erwartungswert auf der rechten Seite ergibt sich durch die zufällige Wahl von A_q gemäß $\mathbf{q}$. Für jede Eingabe I handelt es sich dabei also um die „erwartete Laufzeit“ gewisser deterministischer Algorithmen für I.
- Wir erinnern an Punkt 1.1. Jeder (randomisierte) Las-Vegas-Algorithmus kann als eine Menge deterministischer Algorithmen aufgefasst werden, aus denen nach einer gewissen Wahrscheinlichkeitsverteilung bei jeder Ausführung einer ausgewählt wird. Das Maximum über verschiedene Eingaben des Erwartungswertes auf der rechten Seite ist also die interessierende Größe.

- Mit anderen Worten: $\min_{A \in \mathcal{A}} \mathbf{E}[C(I_p, A)]$ ist eine untere Schranke für Laufzeit des randomisierten Algorithmus (für gewisse Eingaben).

5.19 Bemerkung. Einen Satz analog zu 5.17 kann man auch für Monte-Carlo-Algorithmen beweisen. Hierauf gehen wir nicht weiter ein.

Wir wollen nun die Minimax-Methode auf das Problem der Auswertung von UOB anwenden.

5.20 Als erstes beobachte man, dass wegen

$$\overline{\overline{(x_1 \vee x_2) \wedge (x_3 \vee x_4)}} = \overline{\overline{(x_1 \vee x_2)} \vee \overline{(x_3 \vee x_4)}} = (x_1 \overline{\vee} x_2) \overline{\vee} (x_3 \overline{\vee} x_4)$$

jeder UOB äquivalent auch als Baum dargestellt werden kann, dessen innere Knoten *alle* die NOR-Funktion $\overline{\vee}$ berechnen.

5.21 Ein $\overline{\vee}$-Gatter liefert genau dann eine 1, wenn an beiden Eingängen eine 0 vorliegt.

Die Zahl $p = \frac{3-\sqrt{5}}{2}$ hat die Eigenschaft $(1-p)^2 = p$ (wie man durch einfaches Nachrechnen sieht). Wenn an jedem Eingang eines $\overline{\vee}$-Gatters unabhängig mit Wahrscheinlichkeit p eine 1 vorliegt, ist daher mit gleicher Wahrscheinlichkeit p auch die Ausgabe eine 1.

Als letzten vorbereitenden Schritt benötigen wir noch die folgende Tatsache.

5.22 Satz. *Es sei T ein vollständiger balancierter Baum aus $\overline{\vee}$-Knoten, dessen Blätter alle unabhängig voneinander mit einer Wahrscheinlichkeit q den Wert 1 haben. Es sei $W(T)$ das Minimum (genommen über alle deterministischen Algorithmen) der erwarteten Anzahl von Schritten zur Auswertung von T.*

Dann gibt es auch einen Algorithmus A, der eine erwartete Anzahl von nur $W(T)$ Schritten macht und außerdem die folgende Eigenschaft hat: Besucht A ein Blatt v', das zu einem Teilbaum T' gehört und später ein Blatt u, das nicht *zu T' gehört, dann gilt für alle Blätter a'' von T', die A überhaupt besucht: A besucht a'' vor u.*

Damit können wir nun beweisen:

5.23 Satz. *Die erwartete Anzahl der Blätter, die ein randomisierter Algorithmus zur Auswertung von UOB mit n Blättern besucht, ist mindestens $n^{\log_2((1+\sqrt{5})/2)} = n^{0.694\ldots}$.*

5.24 Beweis. Wir betrachten nun einen Algorithmus wie in Satz 5.22 und die Auswertung von $\overline{\vee}$-Bäumen, deren Blätter unabhängig voneinander mit Wahrscheinlichkeit $p = \frac{3-\sqrt{5}}{2}$ auf 1 gesetzt sind. In Abhängigkeit von der Höhe h sei $W(h)$ die erwartete Anzahl besuchter Blätter.

Offensichtlich ist

$$\begin{aligned} & W(h) &=& W(h-1) + (1-p)W(h-1) = (2-p)W(h-1) \\ \text{also} \quad & W(h) &=& (2-p)^{h-1} W(1) = (2-p)^h \end{aligned}$$

Einsetzen von $h = \log_2 n$ und p ergibt

$$W(T) = W(\log_2 n) = (2-p)^{\log_2 n} = 2^{(\log_2(2-p))(\log_2 n)} = n^{\log_2(2-p)} = n^{0.694\ldots}$$

■

5.25 Durch eine genauere (und schwierigere) Analyse kann man sich davon überzeugen, dass sogar die obere Schranke von $n^{\log_4 3} \approx n^{0.792\cdots}$ aus Satz 5.7 gleichzeitig auch untere Schranke ist. Algorithmus 5.6 ist also optimal.

Zum Abschluss dieses Kapitels wollen wir noch auf einen anderen Aspekt aufmerksam machen, der sich hinter Satz 5.7 verbirgt.

5.26 Da der Erwartungswert für die Anzahl besuchter Blätter $n^{0.792\cdots}$ ist, muss es mindestens eine Berechnung geben, während der höchstens so viele Blätter besucht werden. (Würden stets mehr Blätter besucht, könnte der Erwartungswert nicht so klein sein.)

Oder anders formuliert: Mit einer gewissen Wahrscheinlichkeit echt größer 0 findet der randomisierte Algorithmus eine Teilmenge von höchstens $n^{0.792\cdots}$ Blättern, aus deren Werten bereits der der Wurzel folgt.

Also existiert, und zwar *für jede Eingabe* (i. e. Verteilung von Bits auf alle Blätter), eine solche „kleine“ Teilmenge von Blättern, deren Kenntnis für die Bestimmung des Wertes an der Wurzel ausreicht.

Andererseits haben wir in Satz 5.4 gesehen, dass jeder deterministische Algorithmus für manche Eingaben alle Blätter besuchen muss. Es ist also manchmal „sehr schwierig“, deterministisch eine solche kleine Teilmenge zu finden.

Zusammenfassung

1. Bei der Auswertung von Und-Oder-Bäumen kann man randomisiert weniger Blattbesuche erwarten, als jeder deterministische Algorithmus für manche Bäume durchführen *muss*.
2. Die Minimax-Methode von Yao liefert eine Möglichkeit, untere Schranken für die erwartete Laufzeit randomisierter Algorithmen herzuleiten.

Literatur

Loomis, L. H. (1946). „On a theorem of von Neumann“. In: *Proceedings of the National Academy of Sciences of the USA* 32, S. 213–215 (siehe S. 42).

Neumann, John von (1928). „Zur Theorie der Gesellschaftsspiele“. In: *Mathematische Annalen* 100, S. 295–320 (siehe S. 42).

