9

Bäume und Co-Graphen

Wie bereits in Kapitel 3 erläutert, gibt es zahlreiche spezielle Graphklassen. Lassen sich alle Graphen einer Klasse entlang einer Baumstruktur aufbauen, so spricht man von einer baumstrukturierten Graphklasse. Baumstrukturierte Graphen spielen in der Informatik eine wichtige Rolle, da eine unterliegende Baumstruktur sehr oft eine systematische und effiziente Analyse der entsprechenden Graphen ermöglicht. Voraussetzung für diese effiziente Analyse ist, dass die Baumstruktur der Graphen explizit gegeben ist bzw. effizient bestimmt werden kann. Die algorithmische Grundidee besteht in einer dynamischen Programmierung, bei der alle Teillösungen für die Teilgraphen berechnet werden, die durch Teilbäume der Baumstruktur repräsentiert werden. Die maximale Anzahl der Teillösungen ist oft unabhängig von der Größe der Teilgraphen (bzw. polynomiell in der Größe der Teilgraphen). In diesen Fällen können die entsprechenden Graphenprobleme auf den baumstrukturierten Graphen mit dynamischer Programmierung von unten nach oben (englisch: *bottom-up*) entlang der gegebenen Baumstruktur in linearer Zeit (bzw. in polynomieller Zeit) gelöst werden. Um dies algorithmisch zu realisieren, eignet sich die in Definition 3.12 beschriebene Bottom-up-Reihenfolge, bei der die Knoten eines (geordneten) Wurzelbaums so durchlaufen werden, dass die Kinder vor ihren Vorgängerknoten besucht werden, also zuerst einige Blätter und zuletzt die Wurzel des Baums.

Wir wollen nun den algorithmischen Nutzen spezieller Graphklassen am Beispiel von Bäumen und Co-Graphen veranschaulichen.

9.1 Algorithmen auf Bäumen

9.1.1 Definition und grundlegende Eigenschaften

Die Begriffe *Baum* und *Wurzelbaum* wurden in Definition 3.10 eingeführt. Abbildung 9.1 zeigt zwei ungerichtete Bäume. Es gibt zahlreiche äquivalente Charakterisierungen für Bäume, die leicht bewiesen werden können. So kann man Bäume zum Beispiel durch eindeutige Wege, als minimale zusammenhängende Graphen oder als

F. Gurski et al., *Exakte Algorithmen für schwere Graphenprobleme*, eXamen.press,
DOI 10.1007/978-3-642-04500-4_9,

Abb. 9.1. Zwei Bäume

maximale kreisfreie Graphen charakterisieren (siehe die letzten drei Aussagen in Satz 9.1).

Satz 9.1. *Für jeden Graphen $B = (V,E)$ sind die folgenden Aussagen äquivalent:*

1. *B ist ein Baum.*
2. *B ist kreisfrei und $|E| = |V| - 1$.*
3. *B ist zusammenhängend und $|E| = |V| - 1$.*
4. *Zwischen je zwei Knoten aus B existiert genau ein Weg.*
5. *B ist zusammenhängend und jede Kante aus B ist eine Brücke.*
6. *B ist kreisfrei und durch Einfügen einer Kante zwischen zwei beliebigen nicht adjazenten Knoten entsteht ein Kreis.* **ohne Beweis**

Übung 9.2. Beweisen Sie Satz 9.1.

Es stellt sich die Frage, wozu man diese zahlreichen Charakterisierungen für Bäume benötigt. Eine wichtige Anwendung besteht darin, dass man die folgende Aussage mit einer der angegebenen Charakterisierungen leicht zeigen kann.

Satz 9.3. *Man kann für jeden Graphen $G = (V,E)$ in der Zeit $\mathcal{O}(|V|)$ entscheiden, ob G ein Baum ist.* **ohne Beweis**

Übung 9.4. Beweisen Sie Satz 9.3 unter Verwendung einer geeigneten Charakterisierung aus Satz 9.1.

Der Begriff des perfekten Graphen wurde in Definition 3.50 eingeführt und in Satz 3.53 wurden einige Charakterisierungen perfekter Graphen angegeben.

Lemma 9.5. *Jeder Baum ist ein perfekter Graph.*

Beweis. Wir zeigen die Aussage unter Verwendung der dritten Charakterisierung von perfekten Graphen aus Satz 3.53: Ein Graph ist genau dann perfekt, wenn er für kein $n \geq 2$ den ungerichteten Kreis C_{2n+1} mit $2n+1$ Knoten oder dessen Komplementgraphen $\overline{C_{2n+1}}$ als induzierten Teilgraphen enthält.

Nach Definition können Bäume keine Kreise C_n, $n \geq 3$, als Teilgraphen und somit auch keine Kreise C_{2n+1}, $n \geq 2$, als induzierte Teilgraphen enthalten.

Weiterhin kann man die Komplemente der Kreise C_{2n+1}, $n \geq 2$, wie folgt als induzierte Teilgraphen ausschließen. Da der Kreis C_5 selbstkomplementär (also isomorph zu seinem Komplementgraphen) ist, kann ein Baum keinen $\overline{C_5}$ als induzierten Teilgraphen enthalten. Außerdem enthält jeder Kreis C_n, $n \geq 6$, eine unabhängige Menge der Größe drei. Folglich enthält jedes Komplement eines Kreises C_n, $n \geq 6$, einen Kreis mit drei Knoten als induzierten Teilgraphen. Somit enthalten Bäume keine Kreise $\overline{C_{2n+1}}$, $n \geq 2$, als induzierte Teilgraphen. ❑

9.1.2 Algorithmen

In diesem Abschitt beschreiben wir, wie man für Bäume das in Abschnitt 3.4 definierte Problem UNABHÄNGIGE MENGE, das für allgemeine Graphen NP-vollständig ist, sehr einfach mittels dynamischer Programmierung in linearer Zeit lösen kann.

Es sei $B = (V,E)$ ein Baum. Offenbar ist jede unabhängige Menge von B auch eine unabhängige Menge in jedem Wurzelbaum $B_w = (V,E,w)$, der aus B entsteht, indem ein beliebiger Knoten w aus B als seine Wurzel ausgezeichnet wird; und umgekehrt. Damit folgt für jeden Knoten $w \in V$ die Gleichheit

$$\alpha(B) = \alpha(B_w),$$

wobei $\alpha(G)$ die Unabhängigkeitszahl eines Graphen G bezeichnet. Für $v \in V$ bezeichnen wir mit $B_v = (V',E',v)$ den Teilbaum von $B_w = (V,E,w)$ mit Wurzel v. Als Hilfsvariable benötigen wir zusätzlich noch $\alpha'(B_v)$, die Größe einer kardinalitätsmaximalen unabhängigen Menge im Teilbaum B_v, die den Wurzelknoten v *nicht* enthält. Für den Wurzelbaum $B_w = (V,E,w)$ kann die Größe einer kardinalitätsmaximalen unabhängigen Menge $\alpha(B_w)$ durch einen Durchlauf gemäß der Bottom-up-Reihenfolge (siehe Definition 3.13) der Knoten mit der folgenden Aktion auf den Knoten berechnet werden:

1. Setze für jedes Blatt v in B_w:

$$\begin{aligned} \alpha(B_v) &= 1, \\ \alpha'(B_v) &= 0. \end{aligned}$$

2. Setze für jeden inneren Knoten v in B_w:

$$\begin{aligned} \alpha(B_v) &= \max\left\{1 + \sum_{k \text{ ist Kind von } v} \alpha'(B_k), \sum_{k \text{ ist Kind von } v} \alpha(B_k)\right\}, \qquad (9.1) \\ \alpha'(B_v) &= \sum_{k \text{ ist Kind von } v} \alpha(B_k). \end{aligned}$$

Satz 9.6. *Für jeden Baum $B = (V,E)$ kann die Unabhängigkeitszahl in der Zeit $\mathcal{O}(|V|)$ berechnet werden.*

Beweis. Es seien $B = (V,E)$ ein Baum, $w \in V$ und $B_w = (V,E,w)$ der durch w definierte Wurzelbaum. Die Laufzeitabschätzung ist leicht einzusehen, da in der Bottom-up-Reihenfolge für jeden der $|V|$ Knoten des Baums genau einmal zwei Werte ausgerechnet werden müssen und jeder Wert höchstens einmal weiterverwendet wird. Die Korrektheit folgt, da jeder Knoten v des Baums in die unabhängige Menge aufgenommen werden kann (dem entspricht der erste Term in (9.1): $1 + \sum_{k \text{ ist Kind von } v} \alpha'(B_k)$) sowie auch nicht in die unabhängige Menge aufgenommen werden kann (dem entspricht der zweite Term in (9.1): $\sum_{k \text{ ist Kind von } v} \alpha(B_k)$). ❑

Beispiel 9.7 (Unabhängigkeitszahl für Bäume). Wir wenden den Algorithmus auf den Baum in Abb. 9.2 an. Die Knoten sind hier in einer Bottom-up-Reihenfolge nummeriert, d. h., der i-te Knoten in dieser Reihenfolge wird in Abb. 9.2 mit i bezeichnet, z. B. ist 16 die Wurzel des Baums.

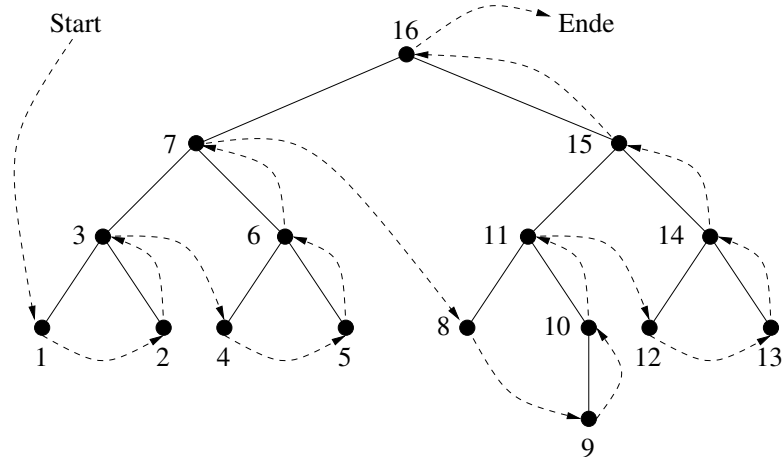

Abb. 9.2. Ein Durchlauf gemäß der Bottom-up-Reihenfolge in einem binären Wurzelbaum

i	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Blatt	+	+	−	+	+	−	−	+	+	−	−	+	+	−	−	−
$\alpha(B_i)$	1	1	2	1	1	2	5	1	1	1	2	1	1	2	5	10
$\alpha'(B_i)$	0	0	2	0	0	2	4	0	0	1	2	0	0	2	4	10

Aufgrund der Beziehung zwischen der Unabhängigkeitszahl und der Knotenüberdeckungszahl in einem Graphen aus Gleichung (3.4) gilt auch die folgende Aussage.

Korollar 9.8. *Für jeden Baum $B = (V,E)$ kann die Knotenüberdeckungszahl in der Zeit $\mathcal{O}(|V|)$ berechnet werden.*

Das Problem PARTITION IN CLIQUEN lässt sich auf das Problem UNABHÄNGIGE MENGE zurückführen, da jeder Baum nach Lemma 9.5 ein perfekter Graph ist.

Korollar 9.9. *Für jeden Baum $B = (V,E)$ kann die Cliquenüberdeckungszahl in der Zeit $\mathcal{O}(|V|)$ berechnet werden.*

Das Problem CLIQUE ist für Bäume sehr einfach zu lösen, da Bäume keine Clique der Größe mindestens drei mehr enthalten können. Das Problem PARTITION IN UNABHÄNGIGE MENGEN ist ebenso einfach, da sich die Knoten eines Baums stets in zwei unabhängige Mengen aufteilen lassen.

Übung 9.10. Wenden Sie die angegebenen Algorithmen auf die beiden Bäume in Abb. 9.1 an, um die folgenden Werte für diese Bäume zu bestimmen:

1. die Unabhängigkeitszahl,
2. die Knotenüberdeckungszahl,
3. die Cliquenzahl,
4. die Färbungszahl und
5. die Cliquenüberdeckungszahl.

Weitere schwere Probleme, die sich in linearer Zeit auf Bäumen lösen lassen, sind PARTITION IN UNABHÄNGIGE KANTENMENGEN und DOMINIERENDE MENGE.

9.2 Algorithmen auf Co-Graphen

9.2.1 Definition und grundlegende Eigenschaften

Zur Definition von Co-Graphen benötigen wir einige Notationen.

Definition 9.11 (disjunkte Vereinigung und disjunkte Summe). *Es seien $G_1 = (V_1,E_1)$ und $G_2 = (V_2,E_2)$ zwei Graphen.*

1. *Die* disjunkte Vereinigung von G_1 und G_2 *ist definiert durch* — disjunkte Vereinigung

$$G_1 \cup G_2 = (V_1 \cup V_2, E_1 \cup E_2). \qquad G_1 \cup G_2$$

2. *Die* disjunkte Summe von G_1 und G_2 *ist definiert durch* — disjunkte Summe

$$G_1 \times G_2 = (V_1 \cup V_2, E_1 \cup E_2 \cup \{\{v_1,v_2\} \mid v_1 \in V_1, v_2 \in V_2\}). \qquad G_1 \times G_2$$

Wie schon in Definition 3.20 erwähnt ist der *Komplementgraph eines Graphen* $G = (V,E)$ definiert durch

$$\overline{G} = (V, \{\{u,v\} \mid u,v \in V, u \neq v, \{u,v\} \notin E\}). \qquad \overline{G}$$

Lemma 9.12. *Die Operationen $\cup$ und $\times$ auf Graphen sind kommutativ, und für je zwei Graphen G_1 und G_2 gilt*

$$G_1 \times G_2 = \overline{\overline{G_1} \cup \overline{G_2}}.$$

ohne Beweis

Übung 9.13. Beweisen Sie Lemma 9.12.

Co-Graphen wurden von verschiedenen Autoren um 1970 unabhängig voneinander eingeführt. Wir wollen hier die folgende Definition verwenden.

Co-Graph **Definition 9.14 (Co-Graph).** *Ein Graph $G = (V,E)$ ist ein* Co-Graph*, falls er sich mit den folgenden drei Regeln aufbauen lässt:*

1. *Der Graph mit genau einem Knoten (kurz bezeichnet mit $G = \bullet$) ist ein Co-Graph.*
2. *Für zwei Co-Graphen G_1 und G_2 ist die disjunkte Vereinigung $G_1 \cup G_2$ ein Co-Graph.*
3. *Für zwei Co-Graphen G_1 und G_2 ist die disjunkte Summe $G_1 \times G_2$ ein Co-Graph.*

Beispiel 9.15 (Co-Graph).

1. Abbildung 9.3 zeigt sechs Co-Graphen.

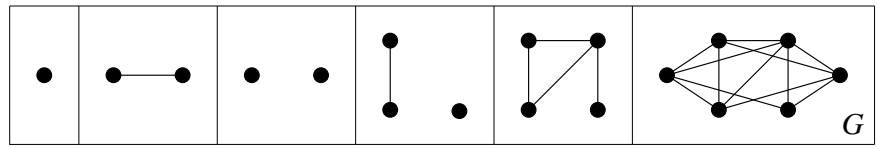

Abb. 9.3. Sechs Co-Graphen

2. Weitere Beispiele für Co-Graphen sind die vollständigen Graphen K_n und die vollständig unzusammenhängenden Graphen $\overline{K_n}$ sowie die vollständig bipartiten Graphen $K_{n,m}$ für alle n und m.

Man kann jeden Aufbau eines Co-Graphen mit den Operationen $\cup$ und $\times$ in einer Baumstruktur darstellen. Ein Co-Baum zu einem Co-Graphen ist ein binärer Wurzelbaum, dessen innere Knoten mit $\cup$ oder $\times$ markiert und dessen Blätter den Knoten des Graphen zugeordnet sind.

Definition 9.16 (Co-Baum).

Co-Baum 1. *Der* Co-Baum T zu einem Co-Graphen $G = \bullet$, der aus genau einem Knoten besteht*, hat genau einen Knoten (die Wurzel von T), der mit $\bullet$ markiert wird.*
2. *Der* Co-Baum T zur disjunkten Vereinigung zweier Co-Graphen G_1 und G_2 *besteht aus der disjunkten Vereinigung der Co-Bäume T_1 und T_2 zu G_1 bzw. G_2, einem zusätzlichen Knoten r (der Wurzel von T), der mit $\cup$ markiert wird, und zwei zusätzlichen Kanten zwischen r und der Wurzel von T_1 bzw. zwischen r und der Wurzel von T_2.*
3. *Der* Co-Baum T zur disjunkten Summe zweier Co-Graphen G_1 und G_2 *ist analog unter Verwendung der disjunkten Summe der Co-Bäume T_1 und T_2 zu G_1 bzw. G_2 definiert.*

Beispiel 9.17 (Co-Baum). Abbildung 9.4 zeigt den Co-Baum T_G zu dem Co-Graphen G, der ganz rechts in Abb. 9.3 dargestellt ist. (Die Zahlen an den Knoten geben eine Bottom-up-Reihenfolge an, die später in den Beispielen 9.28 und 9.31 gebraucht wird.)

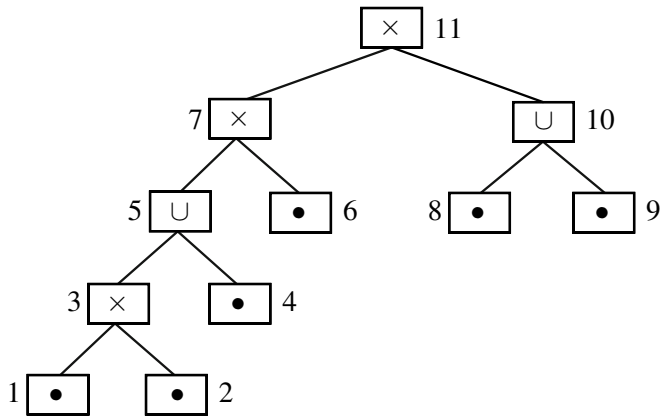

Abb. 9.4. Ein Co-Baum zu dem Co-Graphen G aus Abbildung 9.3

Übung 9.18. In der Literatur werden die Operationen $\cup$ und $\times$ auf Co-Graphen gelegentlich auch für mehr als zwei beteiligte Graphen definiert. Die dadurch entstehenden Co-Bäume sind im Allgemeinen nicht mehr binär. Zeigen Sie, dass man jeden solchen allgemeinen Co-Baum in einen binären Co-Baum transformieren kann, der denselben Co-Graphen definiert.

Übung 9.19. (a) Es sei G ein Co-Graph mit Co-Baum T. Erläutern Sie, wie man aus T einen Co-Baum für den Komplementgraphen von G konstruieren kann.
(b) Zeichnen Sie einen Co-Baum T für den Komplementgraphen des Co-Graphen G aus Abb. 9.3. Geben Sie auch den durch T definierten Co-Graphen $\overline{G}$ an.

Lemma 9.20. *Die Menge der Co-Graphen ist abgeschlossen bezüglich Komplementbildung und induzierter Teilgraphbildung, jedoch nicht abgeschlossen bezüglich Teilgraphbildung.* **ohne Beweis**

Übung 9.21. Zeigen Sie Lemma 9.20.

Im folgenden Satz sind einige Charakterisierungen von Co-Graphen zusammengefasst.

Satz 9.22. *Für jeden Graphen G sind die folgenden Eigenschaften äquivalent.*

1. *G ist ein Co-Graph.*
2. *G enthält keinen P_4 (also keinen Weg mit vier Knoten) als induzierten Teilgraphen. (Kurz: G ist P_4-frei.)*

3. *Der Komplementgraph jedes zusammenhängenden induzierten Teilgraphen mit mindestens zwei Knoten aus G ist unzusammenhängend.*
4. *G lässt sich mit den folgenden drei Regeln aufbauen.*
   a) *Jeder Graph mit genau einem Knoten ist ein Co-Graph.*
   b) *Für zwei Co-Graphen G_1 und G_2 ist die disjunkte Vereinigung $G_1 \cup G_2$ ein Co-Graph.*
   c) *Für einen Co-Graphen G_1 ist auch der Komplementgraph $\overline{G_1}$ ein Co-Graph.*

ohne Beweis

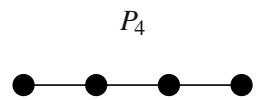

Abb. 9.5. Verbotenener induzierter Teilgraph für Co-Graphen

Übung 9.23. Beweisen Sie, dass der P_4 kein Co-Graph ist, ohne dabei die entsprechende Charakterisierung von Co-Graphen aus Satz 9.22 zu benutzen.

Das folgende Lemma kann einfach bewiesen werden.

Lemma 9.24. *Jeder Co-Graph ist ein perfekter Graph.*

Übung 9.25. Zeigen Sie Lemma 9.24.

Die folgende Aussage ist für die Algorithmen auf Co-Graphen im nächsten Abschnitt wichtig.

Satz 9.26 (Corneil, Perl und Stewart [CPS85]). *Man kann für jeden Graphen $G = (V,E)$ in der Zeit $\mathcal{O}(|V| + |E|)$ entscheiden, ob G ein Co-Graph ist und im positiven Fall auch einen Co-Baum für G bestimmen.* **ohne Beweis**

9.2.2 Algorithmen

Die folgenden Algorithmen zeigen, wie sich die Probleme UNABHÄNGIGE MENGE, CLIQUE, PARTITION IN UNABHÄNGIGE MENGEN und PARTITION IN CLIQUEN auf Co-Graphen sehr einfach lösen lassen.

Es sei G ein Co-Graph und der Wurzelbaum $T = (V_T, E_T, w)$ sei ein Co-Baum für G und habe die Wurzel w. Für einen Knoten $v \in V_T$ sei T_v der Teilbaum von T mit Wurzel v und $G[v]$ sei der durch T_v definierte Co-Graph.

Unabhängige Menge

Für einen Co-Graphen $G = (V,E)$ kann die Größe $\alpha(G) = \alpha(G[w])$ einer kardinalitätsmaximalen unabhängigen Menge mittels dynamischer Programmierung entlang des Co-Baums T durch einen Bottom-up-Durchlauf der Knoten mit der folgenden Aktion auf den Knoten berechnet werden:

1. Für jedes Blatt v in T setze $\alpha(G[v]) = 1$.
2. Setze für jeden mit $\cup$ markierten inneren Knoten v mit den Kindern v_1 und v_2 in T:
$$\alpha(G[v]) = \alpha(G[v_1]) + \alpha(G[v_2]).$$
3. Setze für jeden mit $\times$ markierten inneren Knoten v mit den Kindern v_1 und v_2 in T:
$$\alpha(G[v]) = \max\{\alpha(G[v_1]), \alpha(G[v_2])\}.$$

Satz 9.27. *Für jeden Co-Graphen $G = (V,E)$ kann die Unabhängigkeitszahl in der Zeit $\mathcal{O}(|V| + |E|)$ berechnet werden.*

Beweis. Es seien $G = (V,E)$ ein Co-Graph und T ein Co-Baum für G. Die Laufzeit ergibt sich daraus, dass T genau $|V|$ Blätter und $|V| - 1$ innere Knoten mit jeweils genau zwei Kindern hat und das Bestimmen des Co-Baums die Zeit $\mathcal{O}(|V| + |E|)$ benötigt.

Die Korrektheit im Fall eines mit $\cup$ markierten inneren Knotens v mit den Kindern v_1 und v_2 in T gilt, da der Graph $G[v]$ offensichtlich eine unabhängige Menge der Größe
$$\alpha(G[v_1]) + \alpha(G[v_2])$$
hat. Eine größere unabhängige Menge kann es im Graphen $G[v]$ nicht geben, da sonst einer der beiden beteiligten Teilgraphen eine größere unabhängige Menge als $\alpha(G[v_1])$ bzw. $\alpha(G[v_2])$ haben würde.

Die Korrektheit im Fall eines mit $\times$ markierten inneren Knotens v mit den Kindern v_1 und v_2 in T gilt, da der Graph $G[v]$ eine unabhängige Menge der Größe
$$\max\{\alpha(G[v_1]), \alpha(G[v_2])\}$$
hat. Eine größere unabhängige Menge kann es im Graph $G[v]$ nicht geben, da sonst einer der beiden beteiligten Teilgraphen eine größere unabhängige Menge als $\alpha(G[v_1])$ bzw. $\alpha(G[v_2])$ enthalten würde. ❑

Beispiel 9.28 (Unabhängigkeitszahl für Co-Graphen). Wir wenden den Algorithmus auf den Co-Baum T_G aus Abb. 9.4 für den Co-Graphen G in Abb. 9.3 an. Die Knoten von T_G sind hier in einer Bottom-up-Reihenfolge nummeriert, d. h., i ist der i-te Knoten von T_G in dieser Reihenfolge. Damit entsprechen die Blätter in T_G den Knoten von G und der Knoten 11 entspricht der Wurzel des Co-Baums T_G. Es ergeben sich die folgenden Werte:

i	1	2	3	4	5	6	7	8	9	10	11
Blatt/$\cup$/$\times$	+	+	$\times$	+	$\cup$	+	$\times$	+	+	$\cup$	$\times$
$\alpha(G[i])$	1	1	1	1	2	1	2	1	1	2	2

Die Unabhängigkeitszahl von G ist also $\alpha(G) = 2$.

Aufgrund der Beziehung zwischen der Unabhängigkeitszahl und der Knotenüberdeckungszahl in einem Graphen aus Gleichung (3.4) gilt auch die folgende Aussage.

Korollar 9.29. *Für jeden Co-Graphen $G = (V,E)$ kann die Knotenüberdeckungszahl in der Zeit $\mathcal{O}(|V| + |E|)$ berechnet werden.*

Clique

Für einen Co-Graphen $G = (V,E)$ kann die Größe $\omega(G) = \omega(G[w])$ einer kardinalitätsmaximalen Clique mittels dynamischer Programmierung entlang des Co-Baums T durch einen Bottom-up-Durchlauf der Knoten mit der folgenden Aktion auf den Knoten berechnet werden:

1. Für jedes Blatt v in T setze $\omega(G[v]) = 1$.
2. Setze für jeden mit $\cup$ markierten inneren Knoten v mit den Kindern v_1 und v_2 in T:
$$\omega(G[v]) = \max\{\omega(G[v_1]), \omega(G[v_2])\}.$$
3. Setze für jeden mit $\times$ markierten inneren Knoten v mit den Kindern v_1 und v_2 in T:
$$\omega(G[v]) = \omega(G[v_1]) + \omega(G[v_2]).$$

Analog zum Algorithmus für das Problem UNABHÄNGIGE MENGE ergibt sich aus dem obigen Algorithmus auch für das Problem CLIQUE die folgende Aussage.

Satz 9.30. *Für jeden Co-Graphen $G = (V,E)$ kann die Größe einer kardinalitätsmaximalen Clique in der Zeit $\mathcal{O}(|V| + |E|)$ berechnet werden.*

Beispiel 9.31 (Cliquenzahl für Co-Graphen). Wir wenden den Algorithmus auf den Co-Baum T_G aus Abb. 9.4 für den Co-Graph G in Abb. 9.3 an. Die Co-Baum-Knoten sind hier wieder in einer Bottom-up-Reihenfolge nummeriert (siehe Beispiel 9.28). Hier ergeben sich die folgenden Werte:

i	1	2	3	4	5	6	7	8	9	10	11
Blatt/$\cup$/$\times$	+	+	$\times$	+	$\cup$	+	$\times$	+	+	$\cup$	$\times$
$\omega(G[i])$	1	1	2	1	2	1	3	1	1	1	4

Die Cliquenzahl von G ist also $\omega(G) = 4$.

Alternativ zu der angegebenen Lösung kann man zu einem gegebenen Graphen G mit Co-Baum T den Co-Baum T' für den Komplementgraphen $\overline{G}$ konstruieren und auf diesem das Problem UNABHÄNGIGE MENGE lösen, da gemäß Gleichung (3.3) $\omega(G) = \alpha(\overline{G})$ gilt.

Da Co-Graphen perfekt sind, kann man nun unmittelbar auch die beiden Probleme PARTITION IN UNABHÄNGIGE MENGEN und PARTITION IN CLIQUEN lösen.

Partition in unabhängige Mengen

Für einen Co-Graphen $G = (V,E)$ kann die Färbungszahl $\chi(G) = \chi(G[w])$ mittels dynamischer Programmierung entlang des Co-Baums T durch einen Bottom-up-Durchlauf der Knoten mit der folgenden Aktion auf den Knoten berechnet werden:

1. Für jedes Blatt v in T setze $\chi(G[v]) = 1$.
2. Setze für jeden mit $\cup$ markierten inneren Knoten v mit den Kindern v_1 und v_2 in T:
$$\chi(G[v]) = \max\{\chi(G[v_1]), \chi(G[v_2])\}.$$
3. Setze für jeden mit $\times$ markierten inneren Knoten v mit den Kindern v_1 und v_2 in T:
$$\chi(G[v]) = \chi(G[v_1]) + \chi(G[v_2]).$$

Satz 9.32. *Für jeden Co-Graphen $G = (V,E)$ kann die Färbungszahl in der Zeit $\mathcal{O}(|V| + |E|)$ berechnet werden.*

Beweis. Die Korrektheit dieser Berechnung folgt, da Co-Graphen nach Lemma 9.24 perfekt sind, d. h., es gilt für jeden Co-Graphen G die Gleichheit $\chi(G) = \omega(G)$. ❑

Mit Beispiel 9.31 erhalten wir also für den Co-Graphen G aus Abb. 9.3 eine Färbungszahl von $\chi(G) = \omega(G) = 4$.

Partition in Cliquen

Für einen Co-Graphen $G = (V,E)$ kann die Cliquenpartitionszahl $\theta(G) = \theta(G[w])$ mittels dynamischer Programmierung entlang des Co-Baums T durch einen Bottom-up-Durchlauf der Knoten mit der folgenden Aktion auf den Knoten berechnet werden:

1. Für jedes Blatt v in T setze $\theta(G[v]) = 1$.
2. Setze für jeden mit $\cup$ markierten inneren Knoten v mit den Kindern v_1 und v_2 in T:
$$\theta(G[v]) = \theta(G[v_1]) + \theta(G[v_2]).$$
3. Setze für jeden mit $\times$ markierten inneren Knoten v mit den Kindern v_1 und v_2 in T:
$$\theta(G[v]) = \max\{\theta(G[v_1]), \theta(G[v_2])\}.$$

Satz 9.33. *Für jeden Co-Graphen $G = (V,E)$ kann die Cliquenpartitionszahl in der Zeit $\mathcal{O}(|V| + |E|)$ berechnet werden.*

Beweis. Die Korrektheit dieser Berechnung folgt, da Co-Graphen nach Lemma 9.24 perfekt sind, d. h., es gilt für jeden Co-Graphen G die Gleichheit $\theta(G) = \alpha(G)$. ❑

Mit Beispiel 9.28 erhalten wir also für den Co-Graphen G aus Abb. 9.3 eine Cliquenpartitionszahl von $\theta(G) = \alpha(G) = 2$.

Alternativ kann man zu einem gegebenen Graphen G mit Co-Baum T den Co-Baum T' für den Komplementgraphen $\overline{G}$ konstruieren und auf diesem das Problem PARTITION IN UNABHÄNGIGE MENGEN lösen, da gemäß Gleichung (3.9) $\theta(G) = \chi(\overline{G})$ gilt.

Übung 9.34. Wenden Sie die angegebenen Algorithmen auf den Co-Graphen $G = K_{1,5} \times P_3$ an, um die folgenden Werte für G zu bestimmen:

1. die Unabhängigkeitszahl,
2. die Knotenüberdeckungszahl,
3. die Cliquenzahl,
4. die Färbungszahl und
5. die Cliquenüberdeckungszahl.

Bestimmen Sie dazu zunächst einen Co-Baum zu G.

Hingegen ist das Problem PARTITION IN UNABHÄNGIGE KANTENMENGEN auf Co-Graphen bislang ungelöst.

Wir haben gezeigt, wie man schwere Probleme einfach auf den zwei Graphklassen Bäume und Co-Graphen lösen kann. Diese beiden Graphklassen werden wir in den Kapiteln 10 und 11 verallgemeinern.

9.3 Literaturhinweise

Bäume sind eine der ältesten Graphklassen. Wurzelbäume bilden die Grundlage vieler wichtiger Datenstrukturen in der Informatik.

Co-Graphen wurden zu Beginn der 1970er Jahre von Lerchs eingeführt [Ler71] und sind anders als Bäume vorwiegend theoretisch interessant. Co-Graphen wurden von vielen weiteren Autoren unter zahlreichen Namen definiert, zum Beispiel als D^*-graphs von Jung [Jun78] oder als HD-graphs (hereditary dacey graphs) von Sumner [Sum74]. Die hier betrachteten Algorithmen auf Co-Graphen findet man bereits in der Arbeit von Corneil, Lerchs und Stewart-Burlingham [CLSB81].