7

Exponentialzeit-Algorithmen für Färbbarkeitsprobleme

7.1 Motivation und einfache Ideen

Zwei Algorithmen für Färbbarkeitsprobleme wurden bereits in früheren Kapiteln vorgestellt, nämlich ein Polynomialzeit-Algorithmus für das Zweifärbbarkeitsproblem (siehe Beispiel 2.2 in Abschnitt 2.1) und der naive Algorithmus für das Dreifärbbarkeitsproblem (siehe Beispiel 5.7 in Abschnitt 5.1.2), der in der Zeit

$$\tilde{\mathcal{O}}(3^n) \tag{7.1}$$

arbeitet. In diesem Kapitel wollen wir bessere Algorithmen als den naiven für Dreifärbbarkeit beschreiben und auch solche für andere Färbbarkeitsprobleme.

Zunächst stellen wir in diesem Abschnitt ein paar Algorithmen vor, die auf einfachen Ideen beruhen und den naiven Algorithmus für Dreifärbbarkeit bereits schlagen (auch wenn sie natürlich immer noch Exponentialzeit brauchen; schließlich ist das Dreifärbbarkeitsproblem nach Satz 5.26 NP-vollständig). Anschließend gehen wir kurz auf die Motivation für exakte Exponentialzeit-Algorithmen ein und erläutern, weshalb solche Verbesserungen für praktische Anwendungen sehr sinnvoll sein können.

7.1.1 Erste Idee: Breitensuche

Die $\tilde{\mathcal{O}}(3^n)$-Schranke des naiven Algorithmus für Dreifärbbarkeit aus Beispiel 5.7 kann mit der folgenden einfachen Idee verbessert werden:

1. Wähle einen beliebigen Startknoten des gegebenen Graphen und färbe ihn mit der ersten Farbe.
2. Führe ausgehend vom Startknoten eine Breitensuche auf dem Graphen durch und färbe jeden Knoten anders als den Vorgängerknoten, also mit einer der beiden noch möglichen Farben. Somit gibt es höchstens zwei neue rekursive Aufrufe in jedem Schritt.
3. Gelingt es, den gesamten Graphen in dieser Weise legal mit drei Farben zu färben, so akzeptiere; ist dies irgendwann nicht mehr möglich, so lehne ab.

F. Gurski et al., *Exakte Algorithmen für schwere Graphenprobleme*, eXamen.press,
DOI 10.1007/978-3-642-04500-4_7,

Es ist klar, dass dieser Algorithmus jeden dreifärbbaren Graphen akzeptiert und jeden nicht dreifärbbaren Graphen ablehnt, das Dreifärbbarkeitsproblem also korrekt löst. Da jeder Knoten (außer dem Startknoten), wenn er gefärbt wird, bereits mindestens einen gefärbten Nachbarknoten hat, stehen in jedem Schritt (außer dem ersten) nur höchstens zwei Farben zur Auswahl. Somit ergibt sich eine Laufzeit von

$$\tilde{\mathcal{O}}(2^n). \tag{7.2}$$

7.1.2 Zweite Idee: Auflisten unabhängiger Mengen beschränkter Größe

Zerlegt man die n Knoten des Eingabegraphen in drei Teilmengen, so muss eine kleinste Teilmenge die Größe höchstens $\lceil n/3 \rceil$ haben. Es gibt nicht mehr als $\sum_{i=1}^{\lceil n/3 \rceil} \binom{n}{i}$ solche Mengen. Nun erzeugt man diese der Reihe nach und überprüft jeweils, ob die erzeugte Menge unabhängig ist und ob der Restgraph zweifärbbar (bzw. bipartit) ist. Dieser Test ist in Polynomialzeit möglich. Die Laufzeit des Algorithmus wird also von der Anzahl der Mengen der Größe höchstens $\lceil n/3 \rceil$ dominiert, und es ergibt sich eine Laufzeit von

$$\tilde{\mathcal{O}}(1.8899^n). \tag{7.3}$$

Eine Variation dieser Idee, die eine noch bessere Laufzeit ermöglicht, geht auf Lawler zurück und wird in Abschnitt 7.2 beschrieben.

7.1.3 Dritte Idee: Zufälliges Ausschließen einer Farbe

Die folgende schöne Idee für einen randomisierten Algorithmus für Dreifärbbarkeit wurde von Beigel und Eppstein [BE05] vorgeschlagen: Wähle für jeden der n Knoten zufällig (unter Gleichverteilung) und unabhängig eine Farbe aus, mit der dieser Knoten *nicht* gefärbt werden soll. Mit Wahrscheinlichkeit $1/3$ kann eine solche Zufallsentscheidung falsch sein, woraus folgt, dass die Wahrscheinlichkeit dafür, dass alle n Zufallsentscheidungen mit einer legalen Dreifärbung des Graphen (falls eine existiert) konsistent sind, genau $(2/3)^n$ ist. Hat man diese n Zufallsentscheidungen getroffen, so kann in Polynomialzeit überprüft werden, ob sie gut waren, denn durch die Einschränkung auf höchstens zwei erlaubte Farben pro Knoten kann das Problem zum Beispiel auf 2-SAT reduziert werden.

Randomisierte Algorithmen sind oft schneller als die besten bekannten deterministischen Algorithmen für dasselbe Problem. Natürlich hat dieser Vorteil seinen Preis. Für die Zeitersparnis handelt man sich den Nachteil ein, dass ein randomisierter Algorithmus Fehler machen kann. Um die Fehlerwahrscheinlichkeit möglichst klein zu halten, empfiehlt es sich, einen solchen randomisierten Algorithmus nicht nur einmal auf die Eingabe anzusetzen, sondern in unabhängigen Versuchen so oft mit derselben Eingabe zu wiederholen, bis man insgesamt einen (fest vorgegebenen) exponentiell kleinen Fehler erreicht hat. Bei jedem Versuch werden womöglich andere Zufallsentscheidungen getroffen, und da sich die Fehlerwahrscheinlichkeiten der einzelnen Versuche multiplizieren, wird der Fehler insgesamt mit jedem neuen Versuch kleiner.

Wie viele unabhängige Versuche sollte man machen? Als Faustregel gilt (und man kann dies auch formal beweisen), dass die Anzahl der Versuche (und damit die Gesamtlaufzeit) reziprok zur Erfolgswahrscheinlichkeit des Einzelversuchs sein sollte. Hat ein Durchlauf des Algorithmus bei einer Eingabe der Größe n die Zeitkomplexität $T(n)$ und die Erfolgswahrscheinlichkeit w_n, dann führen für eine vorgegebene Konstante c insgesamt $c \cdot w_n^{-1}$ unabhängige Wiederholungen des Durchlaufs dieses Algorithmus zu einer Fehlerwahrscheinlichkeit von $(1-w_n)^{c/w_n} \leq \mathrm{e}^{-c}$ und zu einer Gesamtlaufzeit von $c \cdot w_n^{-1} \cdot T(n)$.[35] Ist zum Beispiel $c = 100$, $w_n = (2/3)^n$ und $T(n) = n^2$, so ist der Fehler durch

$$\begin{aligned}\mathrm{e}^{-100} &\approx 3.7176 \cdot 10^{-44} \\ &= 0.000\,000\,000\,000\,000\,000\,000\,000\,000\,000\,000\,000\,000\,037\,176\end{aligned}$$

beschränkt, was vernachlässigbar klein ist. Die Laufzeit ergibt sich dann zu $100 \cdot n^2 \cdot (3/2)^n$, wobei wir polynomiale Faktoren vernachlässigen. Demgemäß hat der oben angegebene randomisierte Algorithmus von Beigel und Eppstein eine Laufzeit von

$$\tilde{\mathcal{O}}((2/3)^{-n}) = \tilde{\mathcal{O}}(1.5^n). \tag{7.4}$$

In den Abschnitten 7.3 bis 7.5 wird eine Methode von Beigel und Eppstein [BE05] beschrieben, die auf derselben Idee der geschickten Einschränkung beruht, aber sogar *deterministische* Algorithmen für verschiedene Färbbarkeits- und andere Probleme ermöglicht. Insbesondere schlägt die Laufzeit des deterministischen Dreifärbbarkeitsalgorithmus, der in Abschnitt 7.5 vorgestellt wird, die in (7.4) angegebene Laufzeit des randomisierten Algorithmus von Beigel und Eppstein.

7.1.4 Vierte Idee: Randomisierte lokale Suche

Die folgende Idee für einen randomisierten Algorithmus ist der obigen ganz ähnlich. Diesmal jedoch führen wir eine randomisierte lokale Suche durch. Das bedeutet, dass man an irgendeinem zufällig gewählten Punkt im Lösungsraum startet und hofft, dort eine Lösung gefunden zu haben. Stellt sich diese Hoffnung als trügerisch heraus, so versucht man eine Weile, den ersten, leider erfolglosen Lösungskandidaten zu retten, indem man ihn an zufällig gewählten Stellen „repariert“ – man sucht also lokal in der Nähe des Startpunktes nach einer korrekten Lösung. Gelingt dies nicht innerhalb einer gesetzten Frist, verwirft man diesen Kandidaten als hoffnungslos und wiederholt dieselbe Prozedur mit einem neuen zufällig gewählten Lösungskandidaten.

Ist also ein Graph mit n Knoten gegeben, so rät man anfangs eine (unter Gleichverteilung) zufällige Dreifärbung der Knoten und überprüft, ob sie legal ist. Falls ja, akzeptiert man zufrieden; andernfalls versucht man diese Dreifärbung zu reparieren, indem man für solche Knoten, die einen gleichfarbigen Nachbarknoten haben, zufällig eine der beiden anderen Farben wählt. Insgesamt führt man bis zu $3n$ lokale Reparaturen am aktuellen Lösungskandidaten aus. Wenn man Glück hat, genügt es, diese lokalen Schäden im Lösungskandidaten repariert zu haben, damit der Graph

[35] Dabei ist die Zahl e definiert durch $\mathrm{e} = \lim_{k\to\infty}(1+1/n)^n \approx 2.718281828459$.

nun legal dreigefärbt ist. In diesem Fall akzeptiert man die Eingabe. Es kann aber auch passieren, dass man in diesem ersten Versuch nicht alle Schäden beheben konnte oder dass durch diese Änderungen an anderer Stelle neue Probleme aufgetreten sind.

Ist die Färbung nach $3n$ lokalen Reparaturversuchen am aktuellen Lösungskandidaten immer noch nicht legal, dann war dieser Kandidat so schlecht, dass man ihn wegwerfen und lieber einen ganz neuen probieren sollte. Man verwirft diesen Lösungskandidaten also und startet die Prozedur mit einer wieder zufällig unter Gleichverteilung (und unabhängig von den zuvor getesteten Dreifärbungen) gewählten Dreifärbung des Graphen neu. Da die Erfolgswahrscheinlichkeit eines solchen Durchgangs $(2/3)^n$ beträgt, wie man sich leicht überlegen kann, wiederholt man diese Prozedur so oft, bis man schließlich eine vernünftig kleine Fehlerwahrscheinlichkeit erreicht, die vernachlässigbar ist. Wie im obigen randomisierten Algorithmus für Dreifärbbarkeit von Beigel und Eppstein [BE05] ergibt sich daraus eine Laufzeit von

$$\tilde{\mathcal{O}}((2/3)^{-n}) = \tilde{\mathcal{O}}(1.5^n).$$

7.1.5 Motivation für die Verbesserung von Exponentialzeit-Algorithmen

Die in (7.1), (7.2), (7.3) und (7.4) angegebenen Laufzeiten von Algorithmen für das Dreifärbbarkeitsproblem wurden immer besser: $\tilde{\mathcal{O}}(3^n)$, $\tilde{\mathcal{O}}(2^n)$, $\tilde{\mathcal{O}}(1.8899^n)$ und schließlich $\tilde{\mathcal{O}}(1.5^n)$. In den kommenden Abschnitten werden wir weitere Algorithmen mit noch besseren Schranken für dieses Problem kennen lernen, nämlich $\tilde{\mathcal{O}}(1.4423^n)$ in Satz 7.3 und $\tilde{\mathcal{O}}(1.3289^n)$ in Satz 7.50. Gleichwohl sind all diese Laufzeiten exponentiell. Warum versucht man eigentlich, die Konstante c in einer exponentiellen Komplexitätsschranke $\tilde{\mathcal{O}}(c^n)$ zu verbessern?

Exponentialfunktionen sind asymptotisch schlecht. Exponentialzeit-Algorithmen sind aus Sicht der Praxis nach Möglichkeit zu vermeiden. Und doch ist es der praktische Aspekt, der die intensive Forschung zur Verbesserung exakter Exponentialzeit-Algorithmen motiviert. Zwar werden sich Funktionen wie 3^n und 1.5^n für sehr große n in praktischer Hinsicht nicht unterscheiden, denn ob man viele Jahrmillionen oder nur viele Jahrtausende zur Lösung eines Problems braucht, ist im wirklichen Leben egal. Tot ist man in beiden Fällen längst, wenn der Computer dann endlich die Antwort „42“ gibt.[36]

Aber für kleine n kann es in der Praxis durchaus einen großen Unterschied machen, ob man einen 3^n- oder einen 1.5^n-Algorithmus verwendet. Denn in der Praxis hat man es oft mit moderaten Problemgrößen zu tun, mit Graphen, die dreißig oder hundert Knoten haben zum Beispiel. Auch sind für die Lösung von Problemen in der Praxis typischerweise Fristen gesetzt; mehr als die asymptotische Laufzeit eines Algorithmus interessiert einen dann die absolute Zeit, in der ein Algorithmus eine gegebene Probleminstanz (von moderater Größe) lösen kann.

[36] Sofern es dann noch Computer gibt. Und vorausgesetzt, er ist während dieser extrem langen Rechnung nicht abgestürzt. Und ... so weiter. Siehe [Ada79].

Erinnern wir uns zur Illustration an das in der Einleitung beschriebene Problem des Lehrers, der eine Klassenfahrt organisieren und deshalb ein Graphfärbungsproblem lösen möchte, damit er seine Klasse so auf die Zimmer einer Jugendherberge verteilen kann, dass ihm Ärger möglichst erspart bleibt. Abbildung 1.1 zeigte nur einen Ausschnitt seiner eigentlichen Probleminstanz, denn seine Klasse hat 32 Schülerinnen und Schüler. Nehmen wir an, dass der Lehrer einen Rechner besitzt, der 10^9 Operationen pro Sekunde ausführt. Verwendet er einen 3^n-Algorithmus für sein Problem, so ergibt sich eine absolute Rechenzeit von mehr als 21.4 Tagen, mit einem 1.5^n-Algorithmus wäre er dagegen bereits nach 0.00043 Sekunden fertig. Da die Klassenfahrt bereits übermorgen stattfinden soll, ist es ganz klar, welch großen Unterschied es für den Lehrer zwischen diesen beiden Exponentialzeit-Algorithmen gibt.

Kann man etwa einen 3^n-Algorithmus zu einem Algorithmus mit der Laufzeit $3^{n/3} \approx 1.4423^n$ verbessern, so kann man in derselben absoluten Zeit Eingaben dreifacher Größe bearbeiten, denn $3^{3\cdot(n/3)} = 3^n$. Und dies kann in der Praxis sehr wichtig sein.

Nicht nur in praktischer, sondern auch in theoretischer Hinsicht ist die Verbesserung von Exponentialzeit-Algorithmen von Interesse. Denn erstens erfordern solche Verbesserungen oft neue algorithmische Techniken oder neue Verfahren der Komplexitätsanalyse von Algorithmen, die auch in anderen Zusammenhängen anwendbar sein können. Und schließlich – die Hoffnung stirbt zuletzt! – nähert man sich mit immer kleineren Werten der Konstante c bei einem deterministischen $\tilde{\mathcal{O}}(c^n)$-Algorithmus für ein NP-vollständiges Problem einer Lösung der großen Frage der theoretischen Informatik, der „P = NP?“-Frage: Könnte man irgendwann $c = 1$ erreichen, so hätte man P = NP bewiesen, denn $\tilde{\mathcal{O}}(1^n) = \tilde{\mathcal{O}}(1)$ ist die Klasse der polynomiellen Zeitschranken.

7.2 Berechnung der Färbungszahl mit Lawlers Algorithmus

Algorithmus von Lawler

Einer der ältesten Graphfärbungsalgorithmen ist der Algorithmus von Lawler. In Abschnitt 3.5 wurde erwähnt, dass dieser die Färbungszahl eines gegebenen Graphen (siehe Abschnitt 3.4.2) mittels dynamischer Programmierung bestimmt. Dabei wird die Tatsache benutzt, dass unter allen minimalen legalen Färbungen des Graphen wenigstens eine ist, die eine maximale unabhängige Menge als Farbklasse enthält (siehe Beispiel 3.15 und Übung 3.19 in Kapitel 3). Zur Erinnerung: Eine unabhängige Menge heißt *maximal*, falls sie die Eigenschaft der Unabhängigkeit durch Hinzunahme eines beliebigen Knotens verlieren würde, d. h., falls sie keine echte Teilmenge einer anderen unabhängigen Menge des Graphen ist.

Dynamische Programmierung wird häufig zur Lösung rekursiv definierter Probleme angewandt, wobei die Lösungen kleinerer Teilprobleme, in die man das Ausgangsproblem nach und nach rekursiv zerlegt hat, in einer Tabelle gespeichert und später wiederverwendet werden, um sie zu Lösungen der entsprechenden größeren Teilprobleme aus früheren Rekursionsstufen zusammenzusetzen. Das Speichern die-

ser Werte hat den Vorteil, dass man sie in den verschiedenen Rekursionen nicht immer wieder neu berechnen muss, was beträchtlich Zeit sparen kann.

Ist $G = (V, E)$ der Eingabegraph, so besteht ein Teilproblem darin, die Färbungszahl $\chi(G[U])$ eines durch $U \subseteq V$ induzierten Graphen zu berechnen. Für $U = \emptyset$ ist $G[\emptyset]$ der leere Graph, für den $\chi(G[\emptyset]) = 0$ gilt. Ist $U \neq \emptyset$ und hat man die Färbungszahlen aller induzierten Teilgraphen $G[U']$, $U' \subset U$, bereits bestimmt, so kann man $\chi(G[U])$ gemäß der rekursiven Formel (3.10) (siehe Abschnitt 3.5) bestimmen:

$$\chi(G[U]) = 1 + \min\{\chi(G[U - W])\},$$

wobei über alle maximalen unabhängigen Mengen $W \subseteq U$ minimiert wird.[37] Wie oben erwähnt, liegt dieser Formel die Idee zugrunde, dass es unter allen minimalen legalen Färbungen von G wenigstens eine geben muss, in der eine Farbklasse eine maximale unabhängige Menge von G bildet. Dieser wird eine Farbe zugeordnet, und wir fahren rekursiv mit dem Restgraphen fort.

Der Algorithmus von Lawler berechnet also die Färbungszahlen aller durch maximale unabhängige Teilmengen von V induzierten Graphen mit wachsender Größe, erst für solche Teilmengen von V der Größe eins, dann für solche der Größe zwei und so weiter. In jeder Iteration des Algorithmus sind somit die Färbungszahlen der durch die entsprechenden kleineren Teilmengen induzierten Graphen schon bekannt und können gemäß (3.10) verwendet werden. Insgesamt würde dies für einen gegebenen Graphen G mit n Knoten eine Laufzeit von $\tilde{\mathcal{O}}\left(\sum_{i=1}^{n} \binom{n}{i} 2^i\right)$ ergeben, die in $\tilde{\mathcal{O}}(3^n)$ liegt, denn nach dem binomischen Lehrsatz:

binomischer Lehrsatz

$$\sum_{i=0}^{n} \binom{n}{i} a^i \cdot b^{n-i} = (a+b)^n$$

gilt insbesondere für $a = 2$ und $b = 1$:

$$\sum_{i=0}^{n} \binom{n}{i} 2^i \cdot 1^{n-i} = (2+1)^n = 3^n.$$

Lawler verbessert diese Schranke, indem er zwei Resultate der Graphentheorie anwendet. Einerseits haben Moon und Moser [MM65] bewiesen, dass es höchstens $3^{n/3}$ maximale unabhängige Mengen in einem Graphen mit n Knoten geben kann. Andererseits zeigten Paull und Unger [PU59], wie diese in der Zeit $\mathcal{O}(n^2)$ erzeugt werden können. Dies ergibt eine Laufzeit von

$$\sum_{i=1}^{n} \binom{n}{i} i^2 \cdot 3^{i/3} \leq n^2 \cdot \sum_{i=1}^{n} \binom{n}{i} 3^{i/3} = n^2 \left(1 + 3^{1/3}\right)^n \tag{7.5}$$

für den Algorithmus von Lawler. Die in (7.5) dargestellte Funktion aber ist in $\tilde{\mathcal{O}}(2.4423^n)$, denn $1 + 3^{1/3} < 2.4423$, und dieses Ergebnis halten wir im folgenden Satz fest.

[37] Da für eine maximale unabhängige Teilmenge W einer nicht leeren Menge U stets $W \neq \emptyset$ gilt, ist $U - W$ eine echte Teilmenge von U.

Satz 7.1 (Lawler [Law76]). *Die Färbungszahl eines gegebenen Graphen kann in der Zeit* $\tilde{\mathcal{O}}(2.4423^n)$ *berechnet werden.*

Beispiel 7.2 (Algorithmus von Lawler). Wir wenden den Algorithmus von Lawler auf unser Klassenfahrtsbeispiel an, also auf den Graphen G aus Abb. 1.1. Um Platz beim Schreiben zu sparen und die induzierten Teilgraphen von G übersichtlicher darstellen zu können, haben wir in Abb. 7.1 dabei die Knoten von G umbenannt:

$$v_1 = \text{Paul},\ v_2 = \text{Max},\ v_3 = \text{Edgar},\ v_4 = \text{Moritz},\ v_5 = \text{Steffen}.$$

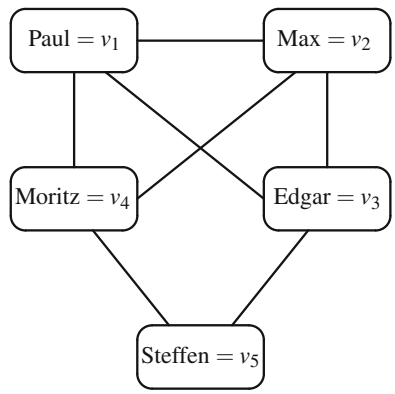


Abb. 7.1. Der Graph aus Abb. 1.1 mit umbenannten Knoten

Da der leere Graph die Färbungszahl null hat, erhalten wir gemäß (3.10) die Färbungszahl eins für jeden induzierten Teilgraphen $G[\{v_i\}]$, $1 \leq i \leq 5$, von G mit nur einem Knoten:

$$\chi(G[\{v_i\}]) = 1 + \chi(G[\emptyset]) = 1 + 0 = 1. \tag{7.6}$$

Bei induzierten Teilgraphen von G, die aus Paaren von Knoten bestehen, ergibt sich:

$$\begin{aligned}
\chi(G[\{v_1,v_2\}]) &= 1 + \min\{\chi(G[\{v_2\}]), \chi(G[\{v_1\}])\} = 1+1 = 2,\\
\chi(G[\{v_1,v_3\}]) &= 1 + \min\{\chi(G[\{v_3\}]), \chi(G[\{v_1\}])\} = 1+1 = 2,\\
\chi(G[\{v_1,v_4\}]) &= 1 + \min\{\chi(G[\{v_4\}]), \chi(G[\{v_1\}])\} = 1+1 = 2,\\
\chi(G[\{v_1,v_5\}]) &= 1 + \min\{\chi(G[\emptyset])\} = 1+0 = 1,\\
\chi(G[\{v_2,v_3\}]) &= 1 + \min\{\chi(G[\{v_3\}]), \chi(G[\{v_2\}])\} = 1+1 = 2,\\
\chi(G[\{v_2,v_4\}]) &= 1 + \min\{\chi(G[\{v_4\}]), \chi(G[\{v_2\}])\} = 1+1 = 2,\\
\chi(G[\{v_2,v_5\}]) &= 1 + \min\{\chi(G[\emptyset])\} = 1+0 = 1,\\
\chi(G[\{v_3,v_4\}]) &= 1 + \min\{\chi(G[\emptyset])\} = 1+0 = 1,\\
\chi(G[\{v_3,v_5\}]) &= 1 + \min\{\chi(G[\{v_5\}]), \chi(G[\{v_3\}])\} = 1+1 = 2,\\
\chi(G[\{v_4,v_5\}]) &= 1 + \min\{\chi(G[\{v_5\}]), \chi(G[\{v_4\}])\} = 1+1 = 2.
\end{aligned}$$

Beispielsweise sind $\{v_1\}$ und $\{v_2\}$ die beiden maximalen unabhängigen Teilmengen von $\{v_1, v_2\}$, da v_1 und v_2 durch eine Kante miteinander verbunden sind. Das Komplement von $\{v_1\}$ in $\{v_1, v_2\}$ ist $\{v_2\}$, und das Komplement von $\{v_2\}$ in $\{v_1, v_2\}$ ist $\{v_1\}$. Die Färbungszahlen der durch $\{v_2\}$ bzw. $\{v_1\}$ induzierten Teilgraphen von G kennen wir jedoch bereits aus (7.6). Es ergibt sich eine Färbungszahl von zwei für $G[\{v_1, v_2\}]$.

Hingegen ist $\{v_1, v_5\}$ selbst die maximale unabhängige Teilmenge von $\{v_1, v_5\}$, da v_1 und v_5 nicht durch eine Kante miteinander verbunden sind. Somit ist das Komplement von $\{v_1, v_5\}$ in $\{v_1, v_5\}$ leer, und die Färbungszahl des leeren Graphen ist null. Es ergibt sich eine Färbungszahl von eins für $G[\{v_1, v_5\}]$. Die übrigen Fälle kann man sich analog überlegen.

Betrachten wir nun in der nächsten Iteration des Lawler-Algorithmus die induzierten Teilgraphen von G, die aus jeweils drei Knoten bestehen, so hängt auch deren Färbungszahl davon ab, welche Kanten es zwischen den jeweiligen drei Knoten gibt, denn dies beeinflusst die Größe der entsprechenden maximalen unabhängigen Teilmengen. Gemäß der rekursiven Formel (3.10) erhalten wir zum Beispiel:

$$\begin{aligned}\chi(G[\{v_1, v_2, v_3\}]) &= 1 + \min\{\chi(G[\{v_2, v_3\}]), \chi(G[\{v_1, v_3\}]), \chi(G[\{v_1, v_2\}])\} \\ &= 1 + \min\{2, 2, 2\} = 3,\end{aligned}$$

weil $\{v_1\}$, $\{v_2\}$ und $\{v_3\}$ die maximalen unabhängigen Mengen in der Clique $\{v_1, v_2, v_3\}$ sind, und deren Komplemente in $\{v_1, v_2, v_3\}$ sind $\{v_2, v_3\}$, $\{v_1, v_3\}$ und $\{v_1, v_2\}$. Die Färbungszahlen dieser Zweiermengen haben wir aber bereits berechnet.

Analog ergeben sich die übrigen Fälle:

$$\begin{aligned}\chi(G[\{v_1, v_2, v_4\}]) &= 1 + \min\{\chi(G[\{v_2, v_4\}]), \chi(G[\{v_1, v_4\}]), \chi(G[\{v_1, v_2\}])\} \\ &= 1 + \min\{2, 2, 2\} = 3, \\ \chi(G[\{v_1, v_2, v_5\}]) &= 1 + \min\{\chi(G[\{v_2\}]), \chi(G[\{v_1\}])\} = 1 + \min\{1, 1\} = 2, \\ \chi(G[\{v_1, v_3, v_4\}]) &= 1 + \min\{\chi(G[\{v_1\}])\} = 1 + 1 = 2, \\ \chi(G[\{v_1, v_3, v_5\}]) &= 1 + \min\{\chi(G[\{v_3\}])\} = 1 + 1 = 2, \\ \chi(G[\{v_1, v_4, v_5\}]) &= 1 + \min\{\chi(G[\{v_4\}])\} = 1 + 1 = 2, \\ \chi(G[\{v_2, v_3, v_4\}]) &= 1 + \min\{\chi(G[\{v_2\}])\} = 1 + 1 = 2, \\ \chi(G[\{v_2, v_3, v_5\}]) &= 1 + \min\{\chi(G[\{v_3\}])\} = 1 + 1 = 2, \\ \chi(G[\{v_2, v_4, v_5\}]) &= 1 + \min\{\chi(G[\{v_4\}])\} = 1 + 1 = 2, \\ \chi(G[\{v_3, v_4, v_5\}]) &= 1 + \min\{\chi(G[\{v_5\}])\} = 1 + 1 = 2.\end{aligned}$$

Betrachten wir nun in der nächsten Iteration des Lawler-Algorithmus die induzierten Teilgraphen von G, die aus jeweils vier Knoten bestehen. Hier erhalten wir aus (3.10) und den bereits ermittelten Färbungszahlen der kleineren induzierten Teilgraphen:

$$\begin{aligned}
\chi(G[\{v_1,v_2,v_3,v_4\}]) &= 1+\min\{\chi(G[\{v_1,v_2\}])\}\\
&= 1+2=3,\\
\chi(G[\{v_1,v_2,v_3,v_5\}]) &= 1+\min\{\chi(G[\{v_2,v_3\}]),\chi(G[\{v_1,v_3\}])\}\\
&= 1+\min\{2,2\}=3,\\
\chi(G[\{v_1,v_2,v_4,v_5\}]) &= 1+\min\{\chi(G[\{v_2,v_4\}]),\chi(G[\{v_1,v_4\}])\}\\
&= 1+\min\{2,2\}=3,\\
\chi(G[\{v_1,v_3,v_4,v_5\}]) &= 1+\min\{\chi(G[\{v_3,v_4\}]),\chi(G[\{v_1,v_5\}])\}\\
&= 1+\min\{1,1\}=2,\\
\chi(G[\{v_2,v_3,v_4,v_5\}]) &= 1+\min\{\chi(G[\{v_3,v_4\}]),\chi(G[\{v_2,v_5\}])\}\\
&= 1+\min\{1,1\}=2.
\end{aligned}$$

Somit erhalten wir in der letzten Iteration des Lawler-Algorithmus als Endresultat:

$$\begin{aligned}
\chi(G) &= 1+\min\{\chi(G[\{v_1,v_2,v_5\}]),\chi(G[\{v_1,v_3,v_4\}]),\chi(G[\{v_2,v_3,v_4\}])\}\\
&= 1+\min\{2,2,2\}=3.
\end{aligned}$$

Der Graph hat also die Färbungszahl drei.

Der Algorithmus von Lawler löst das funktionale Problem (bzw. das Optimierungsproblem), die Färbungszahl eines gegebenen Graphen zu berechnen. Damit kann man natürlich für jedes konstante k das Entscheidungsproblem, ob ein gegebener Graph k-färbbar ist, in der Zeit $\tilde{\mathcal{O}}(2.4423^n)$ lösen. Für $k=3$ liefert der folgende Algorithmus sogar eine bessere Laufzeit:

1. Erzeuge alle maximalen unabhängigen Mengen des gegebenen Graphen.
2. Teste für jede solche Menge, ob ihr Komplement zweifärbbar (also bipartit) ist, und akzeptiere genau dann, wenn eine dieser Mengen den Test besteht.

Dieser Algorithmus ist deutlich besser als der naive Exponentialzeit-Algorithmus für Dreifärbbarkeit (siehe Beispiel 5.7 in Abschnitt 5.1.2), der in der Zeit $\tilde{\mathcal{O}}(3^n)$ arbeitet. Zur Analyse von Lawlers obigem Algorithmus für Dreifärbbarkeit ist lediglich festzustellen, dass

- sämtliche maximalen unabhängigen Mengen eines gegebenen Graphen in Polynomialzeit aufgelistet werden können (siehe [PU59] und auch [JPY88]),
- es nach dem oben genannten Resultat von Moon und Moser [MM65] höchstens $3^{n/3} < 1.4423^n$ maximale unabhängige Mengen in einem Graphen mit n Knoten gibt und
- die Zweifärbbarkeit eines gegebenen Graphen in Polynomialzeit getestet werden kann (siehe Beispiel 2.2 in Abschnitt 2.1).

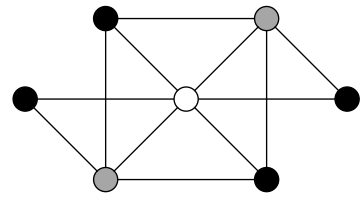
Dieses Ergebnis halten wir hier fest, auch wenn es später – nämlich in Abschnitt 7.5 – noch verbessert werden wird.

Satz 7.3 (Lawler [Law76]). *Das Dreifärbbarkeitsproblem kann in der Zeit*

$$\tilde{\mathcal{O}}(1.4423^n)$$

gelöst werden.

Übung 7.4. Betrachten Sie den dreifärbbaren Graphen G aus Abb. 3.11:



(a) Wenden Sie den Algorithmus von Lawler auf den Graphen G an. Geben Sie dabei wie im Beispiel 7.2 alle Zwischenschritte der Berechnung an.
(b) Schätzen Sie den Platzbedarf des Algorithmus von Lawler ab.

7.3 Constraint Satisfaction

In den folgenden Abschnitten wollen wir die Schranke von $\tilde{\mathcal{O}}(1.4423^n)$ für Dreifärbbarkeit aus Satz 7.3 noch weiter verbessern. Die dafür verwendete Technik ist alles andere als trivial, aber die ihr zugrunde liegende Idee ist ganz simpel und wurde bereits bei dem randomisierten $\tilde{\mathcal{O}}(1.5^n)$-Algorithmus für Dreifärbbarkeit in Abschnitt 7.1.3 erwähnt: Um zu entscheiden, ob ein gegebener Graph dreifärbbar ist oder nicht, ist es nicht nötig, jeden der n Knoten des gegebenen Graphen mit einer von drei Farben zu färben; stattdessen kann man das Problem so einschränken, dass für jeden Knoten nur aus zwei dieser drei Farben eine ausgewählt und die dritte Farbe jeweils verboten wird. Dieses eingeschränkte Problem kann dann in Polynomialzeit gelöst werden, da es aufgrund dieser Einschränkung im Grunde nichts anderes als 2-SAT darstellt.

Genauer gesagt ist der Algorithmus, den wir in Abschnitt 7.4 vorstellen werden, ein rekursiver Verzweigungsprozess, in dessen Verlauf bei jeder Verzweigung entschieden wird, ob für einen Knoten eine von drei Farben verboten oder ob dieser Knoten unmittelbar gefärbt wird. Das Färben eines Knotens unterwirft dabei möglicherweise einige seiner Nachbarn einer Einschränkung, mit welchen Farben sie noch gefärbt werden dürfen, und reduziert die Problemgröße um mehr als einen Knoten.

Es könnte allerdings passieren, dass im Verlauf dieses Algorithmus eine Situation eintritt, in der ungefärbte Knoten von teilweise gefärbten Knoten umgeben sind, sodass sich die Problemgröße nicht schnell genug reduzieren lässt. Diese Schwierigkeit wird dadurch umgangen, dass das Dreifärbbarkeitsproblem als Spezialfall eines allgemeineren Problems aufgefasst wird, einem so genannten *Constraint Satisfaction Problem* (kurz CSP).[38]

Constraint Satisfaction Problem

Zunächst geben wir die allgemeine Definition an.

[38] Wir bleiben hier bei dem englischen Fachbegriff, da dieser auch im Deutschen üblich ist.

Definition 7.5 (Constraint Satisfaction Problem). *Seien a und b zwei positive natürliche Zahlen. Definiere das folgende Entscheidungsproblem:* (a,b)-CSP

(a,b)-CSP	
Gegeben:	*n Variablen, von denen jede einen von bis zu a Werten annehmen kann, und m Einschränkungen, von denen jede ein k-Tupel von Werten für die entsprechenden $k \leq b$ Variablen verbietet.*
Frage:	*Kann man den Variablen Werte so zuweisen, dass jede Einschränkung erfüllt ist, d. h. so, dass jede Einschränkung mindestens eine Variable enthält, der nicht der durch die Einschränkung verbotene Wert zugewiesen wird?*

Da es bei einer (a,b)-CSP-Instanz irrelevant ist, welche a Werte den Variablen zugewiesen werden können, nehmen wir an, dass dies für alle Variablen dieselben Werte sind, nämlich $1,2,\ldots,a$. Statt von „Werten“ werden wir im Folgenden gelegentlich von „Farben“ sprechen, die den Variablen in einer (a,b)-CSP-Instanz zugewiesen werden können – sie werden also mit einer der Farben $1,2,\ldots,a$ „gefärbt“.

Viele Probleme lassen sich in der Form eines (a,b)-CSP ausdrücken. Beispielsweise ist 3-SAT nichts anderes als $(2,3)$-CSP, denn in einer 3-SAT-Instanz können die n Variablen der gegebenen booleschen Formel einen von zwei möglichen Wahrheitswerten (1 für „wahr“ und 0 für „falsch“) annehmen, und jede Klausel der Formel verbietet für die drei in ihr vorkommenden Variablen ein Tripel von Belegungen mit Wahrheitswerten. Ist zum Beispiel die Formel

$$\varphi = (x_1 \vee \neg x_2 \vee x_4) \wedge (\neg x_2 \vee x_3 \vee \neg x_4) \wedge (\neg x_1 \vee x_2 \vee x_4)$$

gegeben, so entspricht

- die Einschränkung $((x_1,0),(x_2,1),(x_4,0))$ der ersten Klausel von φ,
- die Einschränkung $((x_2,1),(x_3,0),(x_4,1))$ der zweiten Klausel von φ und
- die Einschränkung $((x_1,1),(x_2,0),(x_4,0))$ der dritten Klausel von φ.

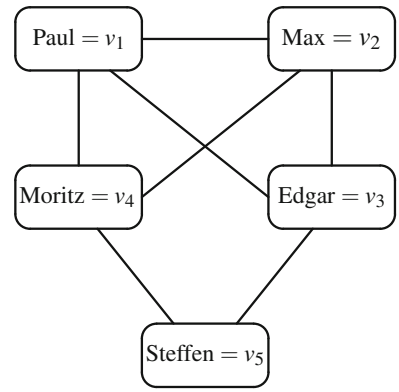
Wie man sieht, ist φ genau dann erfüllbar, wenn sämtliche Einschränkungen der entsprechenden $(2,3)$-CSP-Instanz erfüllt werden können.

Umgekehrt lässt sich jede $(2,3)$-CSP-Instanz durch eine boolesche Formel beschreiben, sodass alle Einschränkungen dieser $(2,3)$-CSP-Instanz genau dann erfüllt werden können, wenn die Formel erfüllbar ist.

Ebenso leicht sieht man, dass das Dreifärbbarkeitsproblem ein Spezialfall des Problems $(3,2)$-CSP ist. Jeder der n Knoten des gegebenen Graphen kann mit einer von drei Farben gefärbt sein, und jede Kante des Graphen entspricht der Einschränkung, dass dieselbe Farbe für die beiden inzidenten Knoten der Kante verboten ist (siehe auch Beispiel 7.6 und Übung 7.7 unten). Analog stellt das Problem k-FÄRBBARKEIT einen Spezialfall von $(k,2)$-CSP dar.

In Abschnitt 7.4 werden wir einen Exponentialzeit-Algorithmus für $(4,2)$-CSP vorstellen, der ebenso für $(3,2)$-CSP funktioniert, und in Abschnitt 7.5 werden wir daraus einen Algorithmus für das Dreifärbbarkeitsproblem ableiten. Zunächst geben wir ein Beispiel für eine $(3,2)$-CSP-Instanz an.

Beispiel 7.6 (Constraint Satisfaction Problem). Betrachten wir wieder den dreifärbbaren Graphen aus Abb. 7.1, der die potentiellen Konflikte zwischen Schülern ausdrückt:



Der Lehrer steht bei der Vorbereitung seiner Klassenfahrt also einer $(a,2)$-CSP-Instanz gegenüber, die er lösen möchte, wobei a die Anzahl der Zimmer in der Jugendherberge ist. Die Frage ist, ob a Zimmer ausreichen, um alle Schüler so unterzubringen, dass sich keine zwei Schüler, die für Stress sorgen könnten, ein Zimmer teilen müssen.

Für $a = 3$ besteht diese $(3,2)$-CSP-Instanz aus den fünf Variablen v_1 (für Paul), v_2 (für Max), v_3 (für Edgar), v_4 (für Moritz) und v_5 (für Steffen), die jeweils mit einer von drei Farben gefärbt werden können (jede Farbe entspricht einer Zimmernummer in der Jugendherberge). Außerdem sind diese Variablen insgesamt 21 Einschränkungen unterworfen (siehe Tabelle 7.1), die sich aus den sieben Kanten des Graphen in Abb. 7.1 ergeben, wobei zu jeder Kante drei Einschränkungen gehören, je eine pro Farbe.

Tabelle 7.1. Einschränkungen der $(3,2)$-CSP-Instanz aus Abb. 7.2

	Einschränkungen der $(3,2)$-CSP-Instanz für		
Kante in G	Farbe 1	Farbe 2	Farbe 3
$\{v_1,v_2\}$	$((v_1,1),(v_2,1))$	$((v_1,2),(v_2,2))$	$((v_1,3),(v_2,3))$
$\{v_1,v_3\}$	$((v_1,1),(v_3,1))$	$((v_1,2),(v_3,2))$	$((v_1,3),(v_3,3))$
$\{v_1,v_4\}$	$((v_1,1),(v_4,1))$	$((v_1,2),(v_4,2))$	$((v_1,3),(v_4,3))$
$\{v_2,v_4\}$	$((v_2,1),(v_4,1))$	$((v_2,2),(v_4,2))$	$((v_2,3),(v_4,3))$
$\{v_2,v_3\}$	$((v_2,1),(v_3,1))$	$((v_2,2),(v_3,2))$	$((v_2,3),(v_3,3))$
$\{v_3,v_5\}$	$((v_3,1),(v_5,1))$	$((v_3,2),(v_5,2))$	$((v_3,3),(v_5,3))$
$\{v_5,v_4\}$	$((v_5,1),(v_4,1))$	$((v_5,2),(v_4,2))$	$((v_5,3),(v_4,3))$

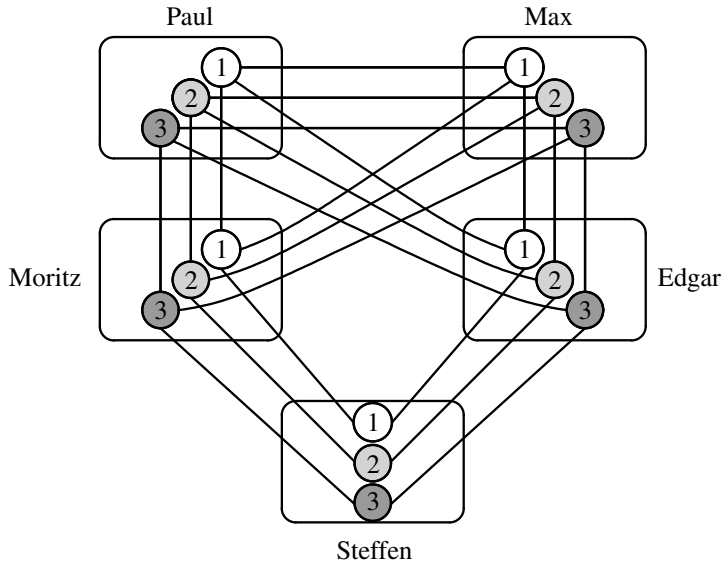


Abb. 7.2. Eine $(3,2)$-CSP-Instanz zum Graphen aus Abb. 7.1

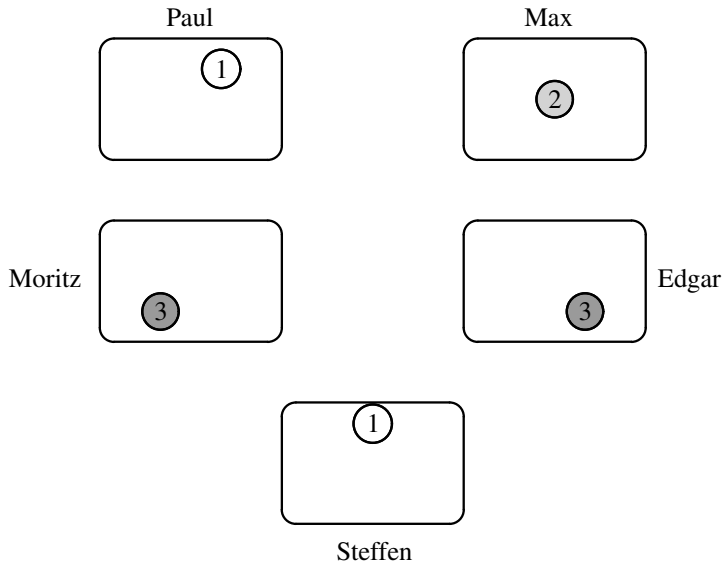


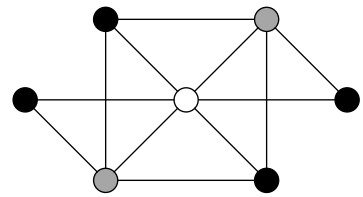
Abb. 7.3. Eine Lösung der $(3,2)$-CSP-Instanz aus Abb. 7.2

Abbildung 7.2 stellt dies graphisch dar. Allerdings darf dieses Bild nicht als ein Graph missverstanden werden, denn durch „Kanten“ verbunden sind hier die Farben innerhalb der Variablenknoten (und nicht die Knoten selbst), die mit den Namen von Schülern markiert sind. Die nummerierten Farben in den Variablenknoten ent-

sprechen also den Zimmern der Jugendherberge, und zwei solche Zimmer sind für verschiedene Schüler in Abb. 7.2 genau dann verbunden, wenn diese beiden Schüler nicht gemeinsam in dieses Zimmer ziehen sollten. Das heißt, die „Kanten“ bzw. Einschränkungen in der Abbildung kennzeichnen für jede Variable die unzulässigen Farbkombinationen mit anderen Variablen.

Das Problem des Lehrers wäre nun gelöst, wenn sich für jeden Schüler (also jede Variable) ein Zimmer (also eine Farbe) finden ließe, sodass alle 21 Einschränkungen erfüllt werden, d. h., wenn in jeder Einschränkung mindestens eine der beiden Zimmerbelegungen nicht realisiert wird. Die in Abb. 7.3 dargestellte Zimmeraufteilung ist eine Lösung der $(3,2)$-CSP-Instanz aus Abb. 7.2, da sie sämtliche Einschränkungen erfüllt. Sie entspricht gerade der Färbung des Graphen G in Abb. 1.2.

Übung 7.7. Stellen Sie den Graphen G aus Abb. 3.11:



als eine $(3,2)$-CSP-Instanz dar und zeigen Sie, dass alle Einschränkungen dieser $(3,2)$-CSP-Instanz erfüllt werden können.

Übung 7.8. Fügen Sie dem Graphen aus Abb. 7.1 noch eine Kante hinzu, entweder eine Kante zwischen Moritz und Edgar oder eine Kante zwischen Paul und Steffen.

(a) Bestimmen Sie in beiden Fällen das kleinste a, sodass die $(a,2)$-CSP-Instanz lösbar ist. Konstruieren Sie diese $(a,2)$-CSP-Instanzen wie in Beispiel 7.6.
(b) Geben Sie in beiden Fällen Lösungen der $(a,2)$-CSP-Instanz an und erläutern Sie, weshalb die jeweiligen $(a-1,2)$-CSP-Instanzen nicht lösbar sind.

Bevor wir im folgenden Abschnitt einen Algorithmus für das Problem $(3,2)$-CSP bzw. $(4,2)$-CSP beschreiben, geben wir ein interessantes Dualitätsresultat an, das beispielsweise zeigt, wie man 3-SAT (das ja äquivalent zu $(2,3)$-CSP ist) auf $(3,2)$-CSP reduzieren kann und wie man umgekehrt $(3,2)$-CSP (und somit insbesondere 3-FÄRBBARKEIT) auf $(2,3)$-CSP (also auf 3-SAT) reduzieren kann.

Satz 7.9 (Beigel und Eppstein [BE05]). *Seien a und b zwei positive natürliche Zahlen. Jede (a,b)-CSP-Instanz I kann in Polynomialzeit in eine (b,a)-CSP-Instanz I' umgewandelt werden kann, sodass gilt:*

$$I \in (a,b)\text{-CSP} \iff I' \in (b,a)\text{-CSP}.$$

Dabei entsprechen die Einschränkungen von I den Variablen in I' und die Einschränkungen von I' den Variablen in I. **ohne Beweis**

Übung 7.10. (a) Beweisen Sie Satz 7.9.

(b) Angenommen, die gegebene (a,b)-CSP-Instanz I hat n Variablen und m Einschränkungen. Schätzen Sie die Größe der resultierenden dualen (b,a)-CSP-Instanz I' ab.

Stattdessen geben wir ein konkretes Beispiel zur Illustration an. Wir werden sehen, dass Variablen und Einschränkungen in den zueinander dualen Instanzen ihre Rollen tauschen. Die Dualität lässt sich daran und an der folgenden Beobachtung erkennen: Während sämtliche Einschränkungen der $(3,2)$-CSP-Instanz aus Beispiel 7.6 (unser Klassenfahrtsbeispiel, siehe insbesondere Tabelle 7.1) das Problem aus Sicht der Jugendherbergszimmer beschreiben (d. h., jede Einschränkung bezieht sich auf dasselbe Zimmer), stellen alle Einschränkungen der dualen $(2,3)$-CSP-Instanz dasselbe Problem in Beispiel 7.11 aus Sicht der Schüler dar (d. h., jede Einschränkung bezieht sich auf denselben Schüler, siehe insbesondere Tabelle 7.2).

Beispiel 7.11 (Dualität gemäß Satz 7.9). Nach Satz 7.9 kann insbesondere eine beliebige $(3,2)$-CSP-Instanz in eine $(2,3)$-CSP-Instanz umgewandelt werden, sodass die angegebenen Eigenschaften gelten, und umgekehrt. Unser Klassenfahrtsproblem, das in Beispiel 7.6 von einer Instanz des Dreifärbbarkeitsproblems in eine $(3,2)$-CSP-Instanz I umgeformt wurde (siehe Abb. 7.2), kann nun weiter in eine $(2,3)$-CSP-Instanz I' übersetzt werden:

1. Die Variablen der zu konstruierenden $(2,3)$-CSP-Instanz I' erhält man wie folgt: Für jede Einschränkung der ursprünglichen $(3,2)$-CSP-Instanz I (siehe Tabelle 7.1 und Abb. 7.2) wird eine neue Variable eingeführt. Insgesamt hat I' dann also 21 Variablen, bezeichnet mit $v_1, v_2, \ldots, v_{21}$, von denen jede mit einer von zwei möglichen Farben gefärbt werden kann:
   a) mit dem linken Paar (V_ℓ, i_ℓ) oder
   b) mit dem rechten Paar (V_r, i_r)

   der entsprechenden Einschränkung $((V_\ell, i_\ell), (V_r, i_r))$ der Originalinstanz I. Dabei sind $V_\ell, V_r \in \{\text{Paul}, \text{Max}, \text{Moritz}, \text{Edgar}, \text{Steffen}\}$ und $i_\ell, i_r \in \{1,2,3\}$. Zum Beispiel gibt es für die Variable, die der Einschränkung $((\text{Paul},1),(\text{Max},1))$ aus Tabelle 7.1 entspricht, die Farben $(\text{Paul},1)$ und $(\text{Max},1)$.
2. Die Einschränkungen von I' bildet man folgendermaßen. Für jedes Tripel von Farben der neuen Variablen in I' (jede Farbe hat ja, wie oben erläutert, die Form (V,i), mit $V \in \{\text{Paul}, \text{Max}, \text{Moritz}, \text{Edgar}, \text{Steffen}\}$ und $i \in \{1,2,3\}$) wird eine Einschränkung von I' so erzeugt, dass alle drei Farben dieser Einschränkung:
   a) dieselbe Originalvariable V von I und
   b) verschiedene Originalfarben für V enthalten,

   d. h., in einer Einschränkung von I' kommen z. B. $(V,1)$, $(V,2)$ und $(V,3)$ vor. Zugeordnet werden diese drei Farben in allen Kombinationen den passenden Variablen v_j von I'.

Abbildung 7.4 zeigt eine Färbung der 21 Variablen der $(2,3)$-CSP-Instanz I', die aus der $(3,2)$-CSP-Instanz I konstruiert wurde. Jede der Variablen v_j von I' enthält zwei Farben – wie oben beschrieben, sind das Paare (V_ℓ, i_ℓ) oder (V_r, i_r) –, und in

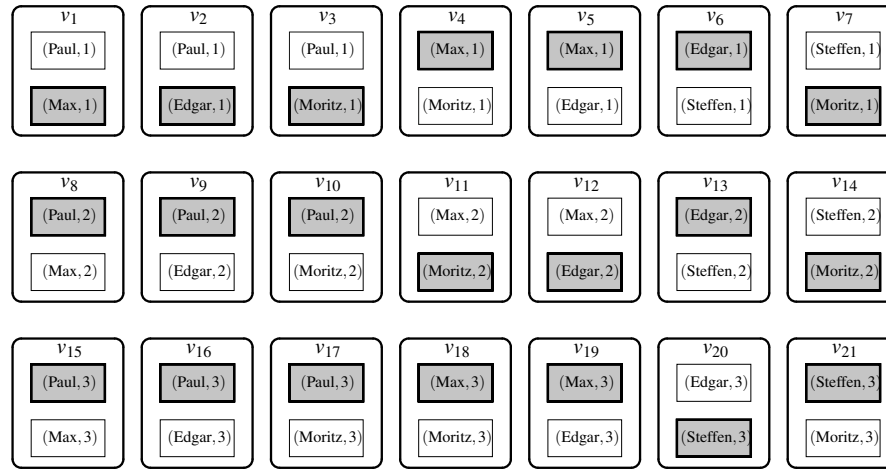


Abb. 7.4. Eine Färbung der 21 Variablen der $(2,3)$-CSP-Instanz I' aus Beispiel 7.11

dieser Färbung wurde v_j mit der fett umrandeten, grau getönten Farbe gefärbt. Intuitiv bedeutet das Färben einer Variablen v_j von I' mit einer Farbe (V,i), dass die Originalfarbe i für die Originalvariable V ausgeschlossen ist.

Wie oben beschrieben, sind die Einschränkungen von I' Tripel von Paaren $(V,1)$, $(V,2)$ und $(V,3)$, die den entsprechenden Variablen v_j zugeordnet werden. Zur Veranschaulichung geben wir in Tabelle 7.2 alle die Einschränkungen für die in Abb. 7.4 dargestellte $(2,3)$-CSP-Instanz I' an, in denen die Originalvariable Paul vorkommt.

Analog erhält man die anderen Einschränkungen, in denen die anderen Originalvariablen (Max, Moritz, Edgar und Steffen) vorkommen. Hinter dieser Konstruktion verbirgt sich die Idee, dass eine Färbung der Variablen der ursprünglichen $(3,2)$-CSP-Instanz I diese genau dann löst, wenn es in jeder Einschränkung von I mindestens ein Paar (V,i) gibt, das in der Färbung nicht vorkommt, d. h., sodass V nicht mit i gefärbt wird. Die oben angegebene Konstruktion bewirkt, dass dies genau dann der Fall ist, wenn jede Einschränkung der resultierenden $(2,3)$-CSP-Instanz I' erfüllt werden kann, weil für jedes V mindestens ein i übrig bleibt, sodass die Farbe (V,i) nicht ausgewählt wird.

Wie man sieht, bläht diese Transformation die Größe der dualen Instanz gehörig auf – grob gesagt exponentiell in dem Parameter a der gegebenen (a,b)-CSP-Instanz I (siehe Übung 7.10(b)). Da a jedoch konstant ist (in unserem Beispiel ist $a=3$), ist die Größe der dualen (b,a)-CSP-Instanz dennoch polynomiell in $|I|$.

Übung 7.12. Betrachten Sie die Formel

$$\varphi = (x_1 \vee \neg x_2 \vee x_4) \wedge (\neg x_2 \vee x_3 \vee \neg x_4) \wedge (\neg x_1 \vee x_2 \vee x_4),$$

die sich wegen 3-SAT $= (2,3)$-CSP auch als eine $(2,3)$-CSP-Instanz auffassen lässt. Wandeln Sie φ gemäß Satz 7.9 in eine äquivalente $(3,2)$-CSP-Instanz um (siehe auch Beispiel 7.11).

Tabelle 7.2. Einschränkungen der $(2,3)$-CSP-Instanz I' aus Abb. 7.4, in denen die Originalvariable Paul vorkommt

$((v_1,(\text{Paul},1)),(v_8,(\text{Paul},2)),(v_{15},(\text{Paul},3)))$
$((v_1,(\text{Paul},1)),(v_8,(\text{Paul},2)),(v_{16},(\text{Paul},3)))$
$((v_1,(\text{Paul},1)),(v_8,(\text{Paul},2)),(v_{17},(\text{Paul},3)))$
$((v_1,(\text{Paul},1)),(v_9,(\text{Paul},2)),(v_{15},(\text{Paul},3)))$
$((v_1,(\text{Paul},1)),(v_9,(\text{Paul},2)),(v_{16},(\text{Paul},3)))$
$((v_1,(\text{Paul},1)),(v_9,(\text{Paul},2)),(v_{17},(\text{Paul},3)))$
$((v_1,(\text{Paul},1)),(v_{10},(\text{Paul},2)),(v_{15},(\text{Paul},3)))$
$((v_1,(\text{Paul},1)),(v_{10},(\text{Paul},2)),(v_{16},(\text{Paul},3)))$
$((v_1,(\text{Paul},1)),(v_{10},(\text{Paul},2)),(v_{17},(\text{Paul},3)))$
$((v_2,(\text{Paul},1)),(v_8,(\text{Paul},2)),(v_{15},(\text{Paul},3)))$
$((v_2,(\text{Paul},1)),(v_8,(\text{Paul},2)),(v_{16},(\text{Paul},3)))$
$((v_2,(\text{Paul},1)),(v_8,(\text{Paul},2)),(v_{17},(\text{Paul},3)))$
$((v_2,(\text{Paul},1)),(v_9,(\text{Paul},2)),(v_{15},(\text{Paul},3)))$
$((v_2,(\text{Paul},1)),(v_9,(\text{Paul},2)),(v_{16},(\text{Paul},3)))$
$((v_2,(\text{Paul},1)),(v_9,(\text{Paul},2)),(v_{17},(\text{Paul},3)))$
$((v_2,(\text{Paul},1)),(v_{10},(\text{Paul},2)),(v_{15},(\text{Paul},3)))$
$((v_2,(\text{Paul},1)),(v_{10},(\text{Paul},2)),(v_{16},(\text{Paul},3)))$
$((v_2,(\text{Paul},1)),(v_{10},(\text{Paul},2)),(v_{17},(\text{Paul},3)))$
$((v_3,(\text{Paul},1)),(v_8,(\text{Paul},2)),(v_{15},(\text{Paul},3)))$
$((v_3,(\text{Paul},1)),(v_8,(\text{Paul},2)),(v_{16},(\text{Paul},3)))$
$((v_3,(\text{Paul},1)),(v_8,(\text{Paul},2)),(v_{17},(\text{Paul},3)))$
$((v_3,(\text{Paul},1)),(v_9,(\text{Paul},2)),(v_{15},(\text{Paul},3)))$
$((v_3,(\text{Paul},1)),(v_9,(\text{Paul},2)),(v_{16},(\text{Paul},3)))$
$((v_3,(\text{Paul},1)),(v_9,(\text{Paul},2)),(v_{17},(\text{Paul},3)))$
$((v_3,(\text{Paul},1)),(v_{10},(\text{Paul},2)),(v_{15},(\text{Paul},3)))$
$((v_3,(\text{Paul},1)),(v_{10},(\text{Paul},2)),(v_{16},(\text{Paul},3)))$
$((v_3,(\text{Paul},1)),(v_{10},(\text{Paul},2)),(v_{17},(\text{Paul},3)))$

7.4 CSP-Algorithmen

In diesem Abschnitt beschreiben und analysieren wir zum einen den deterministischen Algorithmus von Beigel und Eppstein [BE05], der $(3,2)$-CSP in der Zeit $\tilde{\mathcal{O}}(1.36443^n)$ löst (siehe Satz 7.47 in Abschnitt 7.4.3), und wenden ihn in Abschnitt 7.5 auf Färbbarkeitsprobleme wie 3-FÄRBBARKEIT und KANTEN-3-FÄRBBARKEIT an. Zum anderen stellen wir einen randomisierten Algorithmus von Beigel und Eppstein [BE05] vor (siehe Satz 7.22 in Abschnitt 7.4.2), der $(3,2)$-CSP in der erwarteten Zeit $\tilde{\mathcal{O}}(1.4143^n)$ löst, d. h., die Laufzeit dieses Algorithmus – aufgefasst als eine Zufallsvariable bezüglich der Zufallsentscheidungen des Algorithmus – hat diesen Erwartungswert.

erwartete Zeit eines randomisierten Algorithmus

7.4.1 Erste Vereinfachungen

Bevor wir uns dem eigentlichen Algorithmus für $(3,2)$-CSP zuwenden, geben wir im folgenden Lemma ein paar einfache Möglichkeiten an, wie man eine gegebene

$(a,2)$-CSP-Instanz ohne großen Aufwand vereinfachen kann, indem man die Anzahl der Variablen oder der Farben so reduziert, dass die resultierende vereinfachte Instanz äquivalent zur gegebenen Instanz ist. Wir sagen, zwei (a,b)-CSP-Instanzen I und I' sind *äquivalent*, falls I genau dann in (a,b)-CSP ist, wenn I' in (a,b)-CSP ist.

äquivalente (a,b)-CSP-Instanzen

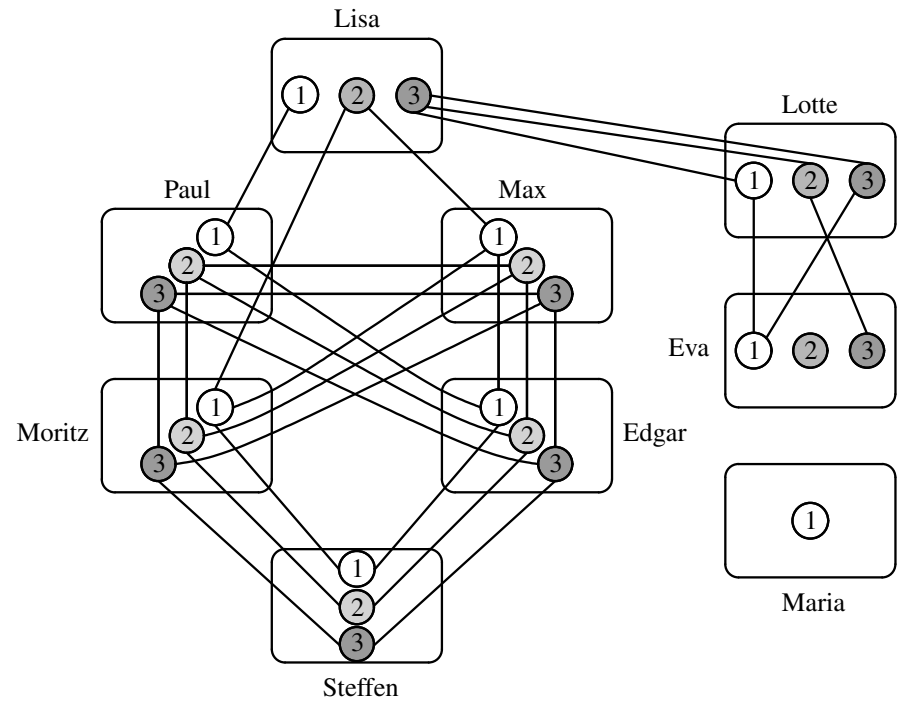


Abb. 7.5. $(3,2)$-CSP-Instanz N zur Illustration von Lemma 7.13

Lemma 7.13 (Beigel und Eppstein [BE05]). *Sei I eine gegebene $(a,2)$-CSP-Instanz.*

1. *Kommt für eine Variable v und eine Farbe i das Paar (v,i) in keiner Einschränkung von I vor, so gibt es eine zu I äquivalente $(a,2)$-CSP-Instanz I' mit einer Variablen weniger als I.*
2. *Sind v eine Variable und i eine Farbe, sodass das Paar (v,i) in Einschränkungen von I mit allen Farbmöglichkeiten einer anderen Variablen w vorkommt, so gibt es eine zu I äquivalente $(a,2)$-CSP-Instanz I', die für eine Variable eine Farbe weniger als I hat.*
3. *Seien v eine Variable und i und j Farben, sodass gilt: Falls I eine Einschränkung $((v,i),(w,k))$ enthält, so enthält I auch eine Einschränkung $((v,j),(w,k))$, $j \neq i$. Dann gibt es eine zu I äquivalente $(a,2)$-CSP-Instanz I', die für eine Variable eine Farbe weniger als I hat.*

4. *Sind u und v verschiedene Variablen und i und j Farben, sodass die einzigen Einschränkungen von I, in denen die Paare (u,i) und (v,j) vorkommen, die Form entweder $((u,i),(v,k))$ mit $k \neq j$ oder $((u,k),(v,j))$ mit $k \neq i$ haben, so gibt es eine zu I äquivalente $(a,2)$-CSP-Instanz I' mit zwei Variablen weniger als I.*
5. *Ist v eine Variable in I, für die nur zwei Farben erlaubt sind, so gibt es eine zu I äquivalente $(a,2)$-CSP-Instanz I' mit einer Variablen weniger als I.*

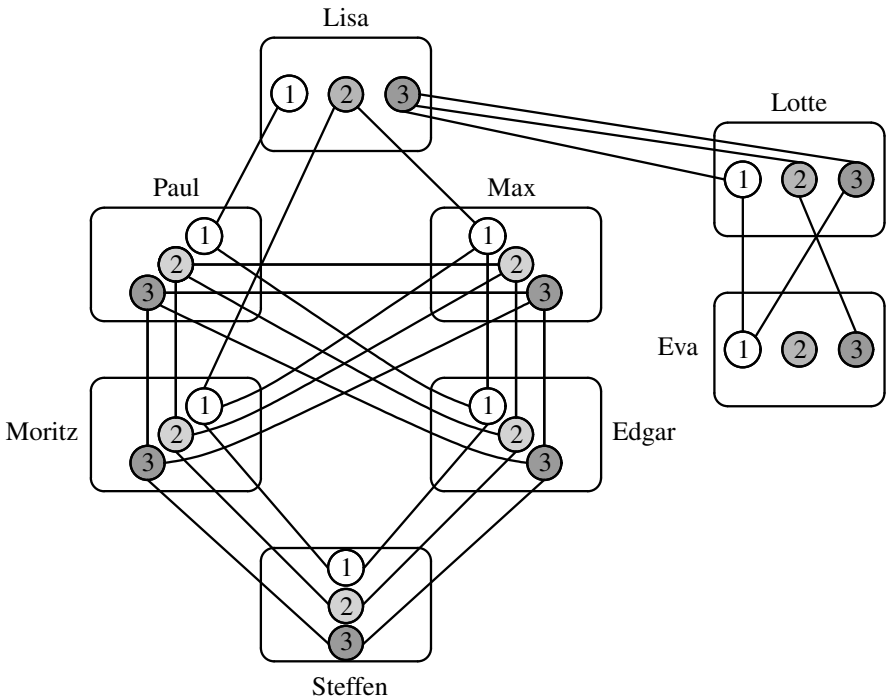


Abb. 7.6. $(3,2)$-CSP-Instanz M, die aus N gemäß Lemma 7.13.1 entsteht

Beweis. Wir illustrieren den einfachen Beweis von Lemma 7.13 anhand der in Abb. 7.5 dargestellten $(3,2)$-CSP-Instanz N, die wir dann gemäß der einzelnen Aussagen des Lemmas Schritt für Schritt umformen und vereinfachen. (Zu beachten ist, dass diese $(3,2)$-CSP-Instanz N nicht mehr im Sinne unseres Klassenfahrtsbeispiels interpretiert werden kann, d. h., sie kann nicht wie in Beispiel 7.6 aus einem Graphen entstanden sein.)

1. Die Variable v kann mit der Farbe i gefärbt und von der Instanz I entfernt werden. Die resultierende Instanz I' ist offensichtlich äquivalent zu I.

   Beispiel 7.14 (für Lemma 7.13.1). Die Variable Maria in der $(3,2)$-CSP-Instanz N aus Abb. 7.5 kommt in keinen Einschränkungen von N vor. Deshalb wird

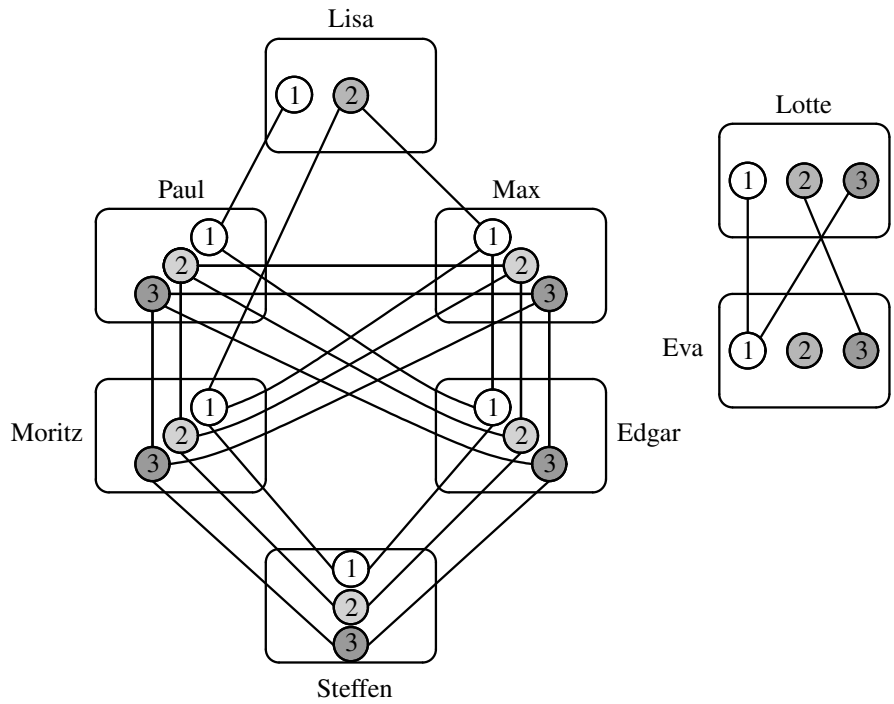


Abb. 7.7. $(3,2)$-CSP-Instanz L, die aus M gemäß Lemma 7.13.2 entsteht

Maria mit Farbe 1 gefärbt[39] und anschließend von der Instanz entfernt[40]. Die resultierende $(3,2)$-CSP-Instanz M ist in Abb. 7.6 dargestellt.

2. Da das Paar (v,i) in den Einschränkungen von I in Kombination mit allen Farben einer anderen Variablen w vorkommt, kann v in keiner Färbung der Variablen von I mit i gefärbt werden, denn andernfalls könnte w gar nicht gefärbt werden. Folglich können wir auf die überflüssige Farbe i für v verzichten. Die resultierende Instanz I' ist offensichtlich äquivalent zu I.

Beispiel 7.15 (für Lemma 7.13.2). Die Farbe 3 der Variablen Lisa ist in der $(3,2)$-CSP-Instanz M aus Abb. 7.6 mit allen drei Farben der Variablen Lot-

[39] Hier sieht man zum Beispiel, weshalb die $(3,2)$-CSP-Instanz N nicht im Sinne des Klassenfahrtsbeispiels interpretiert werden sollte. Würden wir das doch tun, dann kann man sich Steffens Freude vorstellen, wenn Maria in sein Zimmer 1 geschickt wird. Seinen Zimmergenossen Paul – der ja bekanntlich in Lisa verliebt ist – wenigstens zeitweilig loszuwerden, dürfte kein Problem für ihn sein.

[40] Der Lehrer geht auf Nummer sicher. Er kennt Steffen. Und eine völlig uneingeschränkte Schülerin wie Maria könnte er in seiner Probleminstanz sowieso nicht dulden. Aber wie gesagt, wir vermeiden diese in die Irre führenden Fehlinterpretationen, indem wir die $(3,2)$-CSP-Instanzen im Beweis von Lemma 7.13 nicht im Sinne von Klassenfahrt und Zimmerbelegungen auslegen, sondern schlicht und einfach als $(3,2)$-CSP-Instanzen sehen.

te verbunden. Entfernen wir diese Farbe 3 der Variablen Lisa und die entsprechenden drei Einschränkungen, $((\text{Lisa},3),(\text{Lotte},1))$, $((\text{Lisa},3),(\text{Lotte},2))$ und $((\text{Lisa},3),(\text{Lotte},3))$, so erhalten wir die neue $(3,2)$-CSP-Instanz L, die in Abb. 7.7 dargestellt ist.

3. Angenommen, es gilt: Wenn $((v,i),(w,k))$ eine Einschränkung in I ist, so auch $((v,j),(w,k))$, $j \neq i$. Dann kann die Variable v, sollte sie mit der Farbe i gefärbt sein, mit der Farbe j umgefärbt werden, ohne dass andere Einschränkungen verletzt werden. Auf die nun überflüssige Farbe i für v können wir also verzichten. Die resultierende Instanz I' ist offensichtlich äquivalent zu I.

   Beispiel 7.16 (für Lemma 7.13.3). Die Farben 1 und 3 der Variablen Lotte sind in der $(3,2)$-CSP-Instanz L aus Abb. 7.7 mit derselben Farbe 1 der Variablen Eva verbunden. Entfernen wir die Farbe 1 der Variablen Lotte und die entsprechende Einschränkung $((\text{Lotte},1),(\text{Eva},1))$, so erhalten wir die neue $(3,2)$-CSP-Instanz K, die in Abb. 7.8 dargestellt ist.

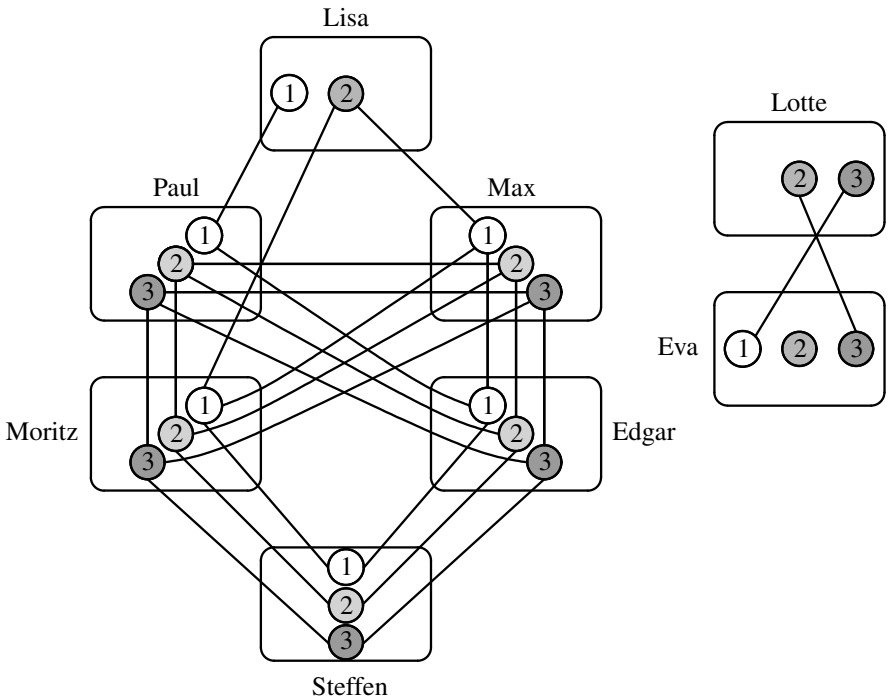


Abb. 7.8. $(3,2)$-CSP-Instanz K, die aus L gemäß Lemma 7.13.3 entsteht

4. Die Variable u kann unbedenklich mit der Farbe i und die Variable v mit der Farbe j gefärbt werden, denn die Voraussetzung dieser Aussage garantiert, dass

es keine Konflikte in den Einschränkungen von I, in denen (u,i) und (v,j) vorkommen, noch irgendwo anders in I gibt. Anschließend können die Variablen u und v entfernt werden, und es liegt auf der Hand, dass die resultierende Instanz I' äquivalent zu I ist.

Beispiel 7.17 (für Lemma 7.13.4). Offenbar sind die einzigen Einschränkungen der $(3,2)$-CSP-Instanz K aus Abb. 7.8, in denen die Paare $(\text{Lotte},2)$ und $(\text{Eva},1)$ vorkommen, $((\text{Lotte},2),(\text{Eva},3))$ und $((\text{Lotte},3),(\text{Eva},1))$. Daher kann man Lotte einfach mit der Farbe 2 und Eva mit der Farbe 1 färben und diese beiden Variablen anschließend entfernen. Abbildung 7.9 zeigt die resultierende neue $(3,2)$-CSP-Instanz J.

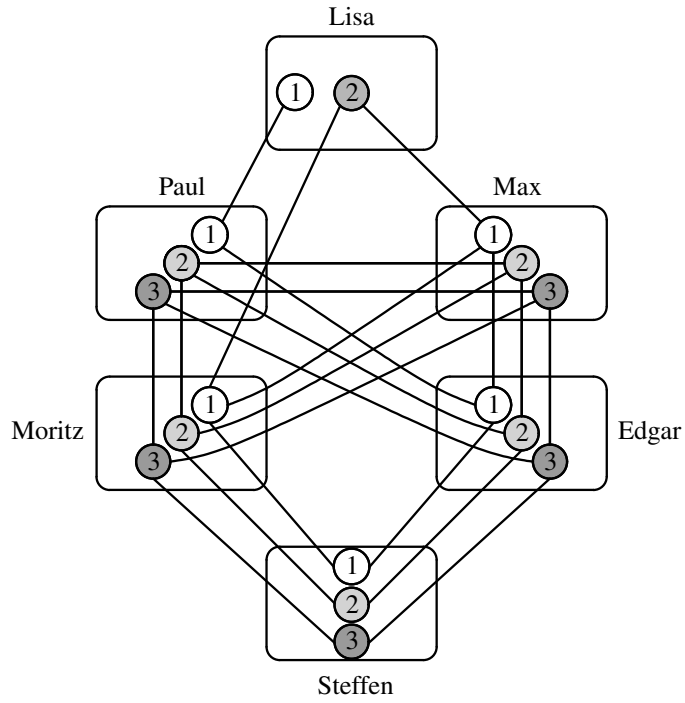


Abb. 7.9. $(3,2)$-CSP-Instanz J, die aus K gemäß Lemma 7.13.4 entsteht

5. Angenommen, nur die Farben 1 und 2 sind für die Variable v erlaubt. Für $i \in \{1,2\}$ definieren wir die Menge

$$\text{Konflikt}(v,i) = \left\{ (w,j) \;\middle|\; \begin{array}{l}(v,i) \text{ und } (w,j) \text{ kommen in} \\ \text{einer Einschränkung von } I \text{ vor}\end{array} \right\}.$$

Fügen wir nun zu den Einschränkungen von I die Menge der Paare

$$\text{Konflikt}(v,1) \times \text{Konflikt}(v,2)$$

hinzu, so kann die Variable v mit allen Einschränkungen, in denen sie vorkommt, bedenkenlos entfernt werden, denn keines der Paare $((w,j),(w',j'))$ in Konflikt$(v,1) \times$ Konflikt$(v,2)$ schließt irgendeine Lösung der ursprünglichen Instanz I aus. Färben wir nämlich w in einer solchen Lösung mit der Farbe j und w' mit der Farbe j', so ist ohnehin keine Farbe für v mehr übrig. Sind umgekehrt alle neuen Einschränkungen in Konflikt$(v,1) \times$ Konflikt$(v,2)$ erfüllt, so muss für v noch eine der beiden Farben 1 und 2 verfügbar sein. Folglich sind die ursprüngliche $(3,2)$-CSP-Instanz I und die aus ihr resultierende $(3,2)$-CSP-Instanz I', die eine Variable weniger hat, äquivalent.

Beispiel 7.18 (für Lemma 7.13.5). Die in Abb. 7.9 dargestellte $(3,2)$-CSP-Instanz J hat eine Variable, Lisa, für die nur zwei Farben erlaubt sind, 1 und 2. Die neuen Einschränkungen sind demnach:

$$\begin{aligned}&\text{Konflikt}(\text{Lisa},1) \times \text{Konflikt}(\text{Lisa},2)\\ &= \{(\text{Paul},1)\} \times \{(\text{Max},1),(\text{Moritz},1)\}\\ &= \{((\text{Paul},1),(\text{Max},1)),((\text{Paul},1),(\text{Moritz},1))\}.\end{aligned}$$

Fügen wir diese zu unserer Instanz J hinzu und entfernen die Variable Lisa und alle Einschränkungen, in denen sie vorkommt, so ergibt sich die zu J äquivalente $(3,2)$-CSP-Instanz I, die in Abb. 7.10 dargestellt (und identisch zu der $(3,2)$-CSP-Instanz aus Abb. 7.2) ist.

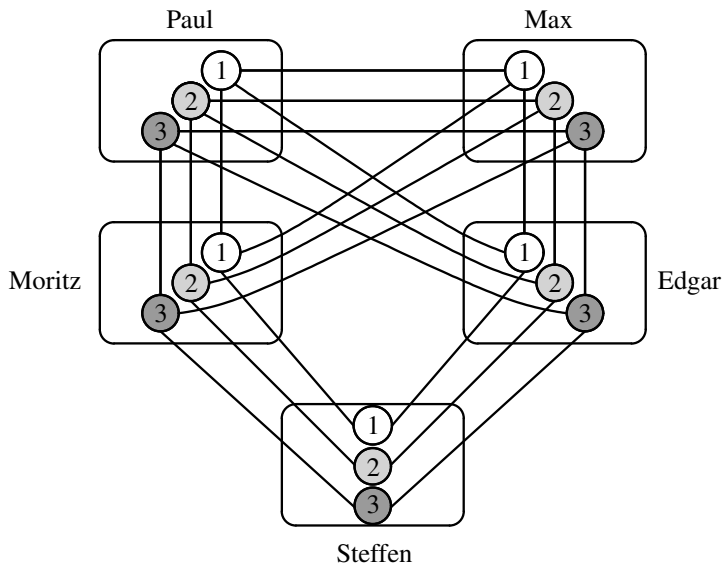


Abb. 7.10. $(3,2)$-CSP-Instanz I, die aus J gemäß Lemma 7.13.5 entsteht

Somit sind alle Aussagen des Lemmas bewiesen. ❑

reduzierte $(a,2)$-CSP-Instanz

Ist keine der Aussagen aus Lemma 7.13 mehr auf eine $(a,2)$-CSP-Instanz anwendbar, so heißt sie *reduziert*. Der deterministische CSP-Algorithmus, den wir in Abschnitt 7.4.3 vorstellen werden, vereinfacht zunächst die gegebene $(3,2)$-CSP- oder $(4,2)$-CSP-Instanz gemäß Lemma 7.13, bis eine reduzierte Instanz vorliegt. Bevor wir uns diesem Algorithmus zuwenden, beschreiben wir jedoch zunächst im folgenden Abschnitt einen einfachen randomisierten Algorithmus, der auf der letzten Aussage von Lemma 7.13 beruht.

7.4.2 Ein randomisierter CSP-Algorithmus

Lemma 7.19 (Beigel und Eppstein [BE05]). *Es gibt einen randomisierten Algorithmus, der eine gegebene* $(3,2)$-CSP-*Instanz* I *in Polynomialzeit in eine* $(3,2)$-CSP-*Instanz* I' *mit zwei Variablen weniger als* I *überführt, sodass gilt:*

1. *Ist* I *lösbar, so ist* I' *mit einer Wahrscheinlichkeit von mindestens* $1/2$ *lösbar, und*
2. *ist* I *nicht lösbar, so ist mit Sicherheit auch* I' *nicht lösbar.*

Beweis. Wenn die gegebene $(3,2)$-CSP-Instanz I überhaupt keine Einschränkung hat, ist sie unmittelbar lösbar. Andernfalls sei $((u,i),(v,j))$ eine beliebige Einschränkung von I. Falls nötig, benennen wir die Farben 1, 2 und 3 der Variablen u und v so um, dass in dieser Einschränkung dieselbe Farbe auftaucht, etwa $i = 1 = j$. Abbildung 7.11 zeigt einen Ausschnitt der Instanz I mit den beiden Variablen u und v und dieser ausgewählten Einschränkung $((u,1),(v,1))$. Die anderen Einschränkungen von I, in denen u und v vorkommen, sind mit dünneren Linien angedeutet.

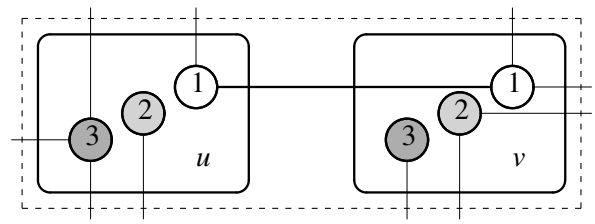
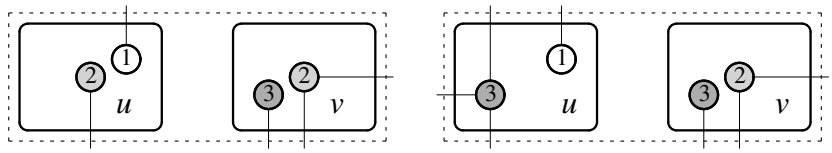


Abb. 7.11. $(3,2)$-CSP-Instanz I im Beweis von Lemma 7.19

Der Algorithmus bringt nun seine Fähigkeit, Münzen werfen, also Zufallsentscheidungen treffen zu können, ins Spiel. Zufällig unter Gleichverteilung wählt er eine der folgenden vier Möglichkeiten, die Anzahl der Farben der Variablen u und v von drei auf zwei zu reduzieren:

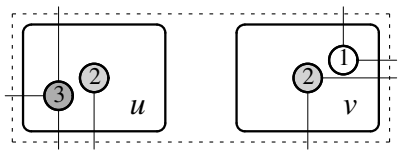
(a) u hat die Farben 1 und 2 und v hat die Farben 2 und 3.
(b) u hat die Farben 1 und 3 und v hat die Farben 2 und 3.
(c) u hat die Farben 2 und 3 und v hat die Farben 1 und 2.
(d) u hat die Farben 2 und 3 und v hat die Farben 1 und 3.

Das heißt, in jeder dieser vier Möglichkeiten gibt es für genau eine der Variablen u und v die Farben 2 und 3. Die entsprechenden Ausschnitte der so modifizierten Instanz sind in Abb. 7.12 dargestellt.

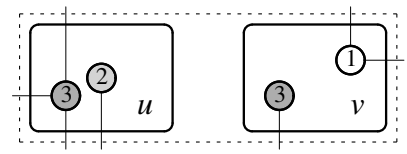


(a) u hat 1 und 2 und v hat 2 und 3

(b) u hat 1 und 3 und v hat 2 und 3



(c) u hat 2 und 3 und v hat 1 und 2



(d) u hat 2 und 3 und v hat 1 und 3

Abb. 7.12. Vier modifizierte $(3,2)$-CSP-Instanzen mit nur zwei Farben für u und v

Angenommen, die ursprüngliche $(3,2)$-CSP-Instanz I ist lösbar. Dann gibt es eine Färbung der Variablen von I, sodass in jeder Einschränkung von I mindestens eines der beiden Paare in der Färbung nicht vorkommt. Insbesondere färbt diese Färbung u oder v nicht mit 1. Für genau zwei der vier in Abb. 7.12 dargestellten reduzierten Instanzen bleibt diese Färbung also gültig und erfüllt alle Einschränkungen der so modizierten Instanz. Da die Gültigkeit einer Lösung von I mit Wahrscheinlichkeit genau $1/2$ für I' erhalten bleibt (es aber auch noch zusätzliche Lösungen für I' geben könnte), ist I' mit einer Wahrscheinlichkeit von mindestens $1/2$ lösbar.

Ist jedoch die ursprüngliche $(3,2)$-CSP-Instanz I nicht lösbar, so kann keine der vier reduzierten Instanzen aus Abb. 7.12 lösbar sein. Um die Kontraposition dieser Implikation zu zeigen, nehmen wir an, I' sei irgendeine reduzierte Instanz, die gemäß einer der Abbildungen 7.12(a) bis 7.12(d) aus I entstanden ist. Ist I' lösbar, so muss diese Lösung auch eine Lösung von I sein, denn die in I' fehlenden Farben der Variablen können nicht Bestandteil der Lösung sein. Also sind alle zusätzlichen Einschränkungen von I erfüllbar.

Auf die zufällig gewählte reduzierte Instanz (siehe Abb. 7.12), die nur noch zwei Farben für die Variablen u und v enthält, kann nun zweimal die Transformation aus der letzten Aussage von Lemma 7.13 angewandt werden, zunächst auf die Variable u, dann auf v. Somit ergibt sich eine äquivalente $(3,2)$-CSP-Instanz I', die der Algorithmus ausgibt. Aus den obigen Betrachtungen folgt, dass I' mit einer Wahrscheinlichkeit von mindestens $1/2$ lösbar ist, falls I lösbar ist, und dass auch I' nicht lösbar ist, falls I nicht lösbar ist. ❑

Beispiel 7.20 (für Lemma 7.19). Betrachten wir wieder die $(3,2)$-CSP-Instanz I aus unserem Klassenfahrtsbeispiel, siehe Abb. 7.13. Der Algorithmus aus Lemma 7.19

ist ausgesprochen natürlich, denn ein Lehrer würde – ohne diesen Algorithmus zu kennen und nur von seiner Intuition geleitet – möglicherweise ganz ähnlich bei der Zimmerzuteilung vorgehen. Das heißt, er würde bei der Lösung seines Problems vielleicht eine beliebige Einschränkung wählen und den zugehörigen Schülern eines von drei Zimmern verbieten, sodass er diese Einschränkung als erledigt betrachten und so sein Problem vereinfachen kann. Tut er dies im Sinne von Abb. 7.12, so kommt er einer Lösung seines Problems mit einer Wahrscheinlichkeit von mindestens $1/2$ näher.

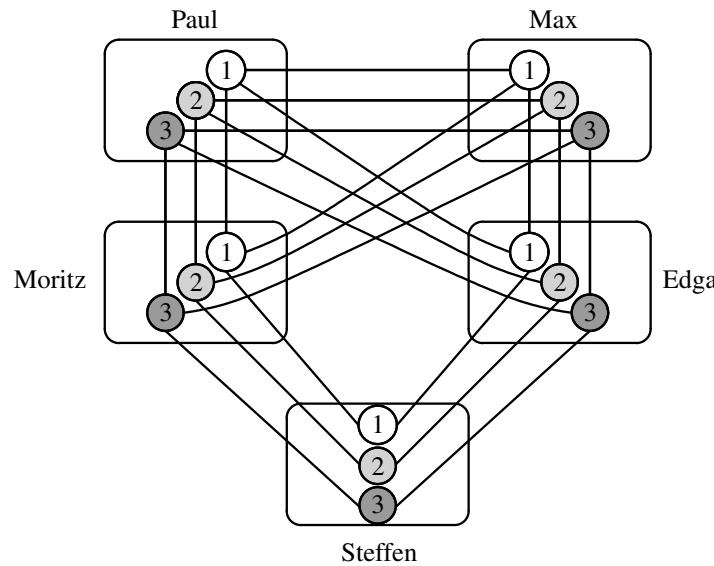


Abb. 7.13. $(3,2)$-CSP-Instanz I für Beispiel 7.20

Nun beschreiben wir dieses Vorgehen etwas formaler. Um die Instanz I gemäß Lemma 7.19 zu reduzieren, wählen wir zunächst eine beliebige Einschränkung, zum Beispiel $((\text{Paul},1),(\text{Max},1))$, und transformieren I gemäß einer der vier Wahlmöglichkeiten, etwa gemäß der aus Abb. 7.12(a). Das heißt, für Paul wird die Farbe 3 und für Max die Farbe 1 ausgeschlossen, und die den ausgeschlossenen Farben entsprechenden Einschränkungen entfallen. Es wird dabei nicht gesagt, welche Farbe (1 oder 2) Paul und welche Farbe (2 oder 3) Max zugewiesen wird, sondern nur gesichert, dass das Färben mit einer dieser Farben später konfliktfrei möglich ist. Die so modifizierte $(3,2)$-CSP-Instanz ist in Abb. 7.14 zu sehen.

Um nun die Variable Paul ganz loszuwerden, also aus unserer Probleminstanz zu entlassen, müssen alle noch vorhandenen Einschränkungen Pauls mit anderen Variablen in geeigneter Weise auf neue Einschränkungen zwischen diesen verbleibenden Variablen übertragen werden. Da für Paul aber nur noch zwei Farben möglich sind, ist dies mit der letzten Aussage von Lemma 7.13 möglich. Formal ergibt sich gemäß dieser Konstruktion:

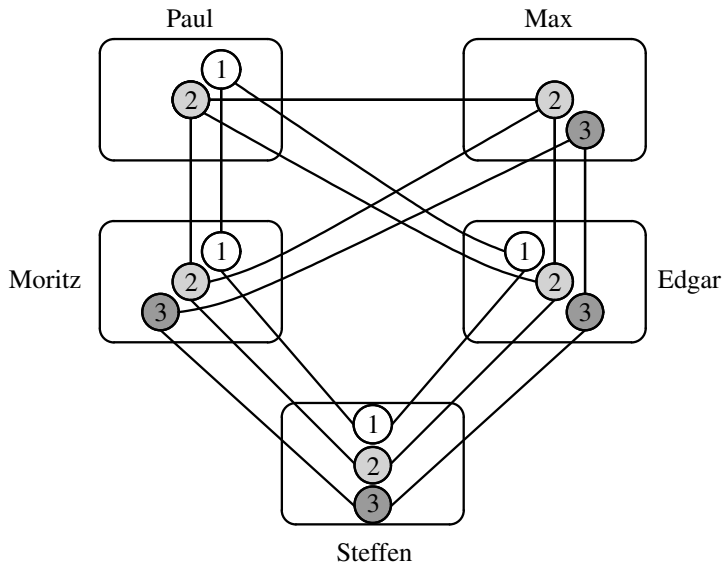


Abb. 7.14. Gemäß Abb. 7.12(a) modifizierte $(3,2)$-CSP-Instanz

$$\begin{aligned}
&\text{Konflikt}(\text{Paul},1) \times \text{Konflikt}(\text{Paul},2)\\
&= \{(\text{Moritz},1),(\text{Edgar},1)\} \times \{(\text{Max},2),(\text{Moritz},2),(\text{Edgar},2)\}\\
&= \{((\text{Moritz},1),(\text{Max},2)),((\text{Moritz},1),(\text{Moritz},2)),((\text{Moritz},1),(\text{Edgar},2)),\\
&\quad ((\text{Edgar},1),(\text{Max},2)),((\text{Edgar},1),(\text{Moritz},2)),((\text{Edgar},1),(\text{Edgar},2))\}.
\end{aligned}$$

Da man jedoch Einschränkungen wie $((\text{Moritz},1),(\text{Moritz},2))$, die dieselbe Variable enthalten, natürlich einfach weglassen kann (denn diese sind immer – in jeder Färbung – erfüllbar, sodass sich dadurch nichts an der Lösbarkeit der Instanz ändert), fügen wir nach dem Entfernen von Paul lediglich die folgenden vier Einschränkungen hinzu:

$$\begin{aligned}
&\{((\text{Moritz},1),(\text{Max},2)),((\text{Moritz},1),(\text{Edgar},2)),\\
&((\text{Edgar},1),(\text{Max},2)),((\text{Edgar},1),(\text{Moritz},2))\}.
\end{aligned}$$

Abbildung 7.15 zeigt die resultierende $(3,2)$-CSP-Instanz.

Mit derselben Prozedur können wir uns auch der Variablen Max entledigen, wieder durch Anwendung der letzten Aussage von Lemma 7.13. Diesmal reduziert sich:

$$\begin{aligned}
&\text{Konflikt}(\text{Max},2) \times \text{Konflikt}(\text{Max},3)\\
&= \{(\text{Moritz},1),(\text{Moritz},2),(\text{Edgar},1),(\text{Edgar},2)\} \times \{(\text{Moritz},3),(\text{Edgar},3)\}\\
&= \{((\text{Moritz},1),(\text{Moritz},3)),((\text{Moritz},1),(\text{Edgar},3)),((\text{Moritz},2),(\text{Moritz},3)),\\
&\quad ((\text{Moritz},2),(\text{Edgar},3)),((\text{Edgar},1),(\text{Moritz},3)),((\text{Edgar},1),(\text{Edgar},3)),\\
&\quad ((\text{Edgar},2),(\text{Moritz},3)),((\text{Edgar},2),(\text{Edgar},3))\}
\end{aligned}$$

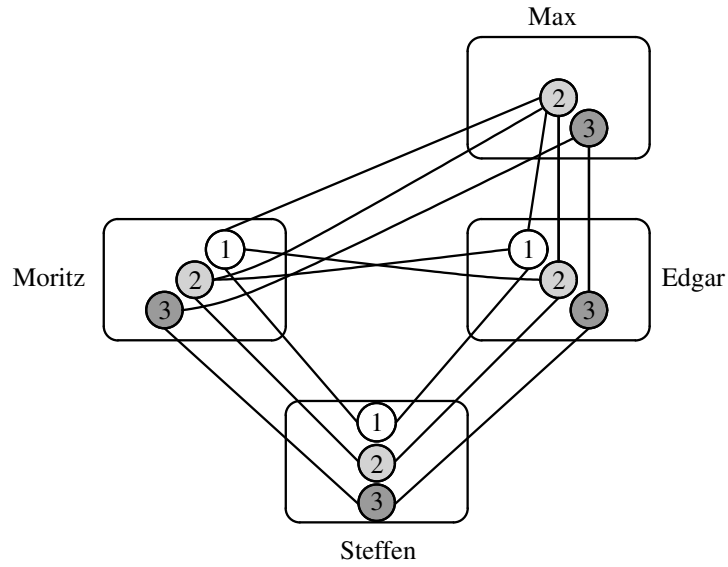


Abb. 7.15. $(3,2)$-CSP-Instanz nach Entfernung von Paul

nach dem Entfernen von Max auf die folgenden vier neuen Einschränkungen:

$$\{((\text{Moritz},1),(\text{Edgar},3)),((\text{Moritz},2),(\text{Edgar},3)),$$
$$((\text{Edgar},1),(\text{Moritz},3)),((\text{Edgar},2),(\text{Moritz},3))\}.$$

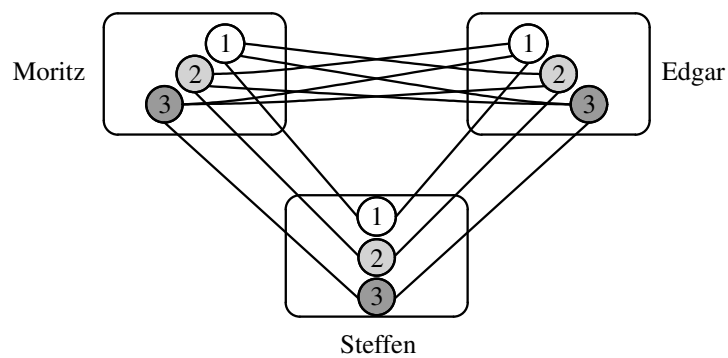


Abb. 7.16. $(3,2)$-CSP-Instanz nach Entfernung von Max

Abbildung 7.16 zeigt die resultierende $(3,2)$-CSP-Instanz. Wie man sieht, ist die Färbung, die die ursprüngliche $(3,2)$-CSP-Instanz I aus Abb. 7.13 erfüllt, eingeschränkt auf Moritz, Edgar und Steffen, immer noch gültig: Färbt man Moritz und Edgar mit der Farbe 3 (beide müssen wegen ihrer Einschränkungen gleich gefärbt

werden) und Steffen mit der Farbe 1, so wird jede Einschränkung der modifizierten $(3,2)$-CSP-Instanz in Abb. 7.16 erfüllt. Dasselbe Ergebnis hätten wir erhalten, wenn wir für die Einschränkung $((\text{Paul},1),(\text{Max},1))$ statt der in Abb. 7.12(a) dargestellten Variante die aus Abb. 7.12(b) gewählt hätten. Mit den Varianten aus Abb. 7.12(c) und Abb. 7.12(d) dagegen ließe sich die ursprüngliche Färbung von I (d. h. $(\text{Paul},1)$, $(\text{Max},2)$, $(\text{Moritz},3)$, $(\text{Edgar},3)$ und $(\text{Steffen},1)$) auf das reduzierte Problem gar nicht übertragen, da dann die Farbe 1 für Paul nicht mehr vorkommt (siehe Übung 7.21). Das schließt jedoch nicht aus, dass es eine andere gültige Färbung für die reduzierte Instanz geben kann, die keine Einschränkung der ursprünglichen Färbung von I ist. Gemäß Lemma 7.19 ist ja die Wahrscheinlichkeit dafür, dass sich die Lösbarkeit der gegebenen $(3,2)$-CSP-Instanz I auf die um zwei Variablen reduzierte $(3,2)$-CSP-Instanz überträgt, *mindestens* $1/2$.

Übung 7.21. Betrachten Sie die in Abb. 7.13 dargestellte $(3,2)$-CSP-Instanz I für Beispiel 7.20.

(a) In Beispiel 7.20 wurden für die Variable Paul die Farbe 3 und für die Variable Max die Farbe 1 ausgeschlossen, d. h., die Zufallswahl entsprach der Modifikation aus Abb. 7.12(a). Rechnen Sie das Beispiel für die drei Fälle durch, in denen die Instanz gemäß
- Abb. 7.12(b),
- Abb. 7.12(c) bzw.
- Abb. 7.12(d)

modifiziert wird. Geben Sie an, ob – und falls ja, wie – die resultierenden Instanzen lösbar sind.

(b) Angenommen, in der Instanz aus Abb. 7.13 für Beispiel 7.20 wird anfangs nicht die Einschränkung $((\text{Paul},1),(\text{Max},1))$, sondern $((\text{Moritz},2),(\text{Steffen},2))$ gewählt. Rechnen Sie das Beispiel für diesen Fall durch, wobei Sie die nötigen Zufallsentscheidungen frei treffen dürfen.

(c) Geben Sie eine lösbare $(3,2)$-CSP-Instanz I an, sodass die gemäß Lemma 7.19 konstruierte Instanz I' mit Wahrscheinlichkeit *genau* $1/2$ lösbar ist.

Satz 7.22 (Beigel und Eppstein [BE05]). $(3,2)$-CSP *kann durch einen randomisierten Algorithmus in der erwarteten Zeit* $\tilde{\mathcal{O}}(1.4143^n)$ *gelöst werden.*

Beweis. Ist eine $(3,2)$-CSP-Instanz mit n Variablen gegeben, muss die in Lemma 7.19 beschriebene Reduktion nur $(n/2)$-mal ausgeführt werden, um eine trivial lösbare Instanz zu erhalten. Das ergibt insgesamt einen randomisierten Algorithmus, der in Polynomialzeit arbeitet und eine Erfolgswahrscheinlichkeit von mindestens $(1/2)^{n/2} = 2^{-n/2}$ hat. Nach der in Abschnitt 7.1.3 genannten Faustregel, dass die Anzahl der wiederholten Versuche reziprok zur Erfolgswahrscheinlichkeit des Einzelversuchs sein sollte, folgt, dass nach $2^{n/2}$ Versuchen eine korrekte Lösung erwartet werden kann. Somit hat dieser randomisierte Algorithmus eine erwartete Laufzeit von $\tilde{\mathcal{O}}(2^{n/2}) \approx \tilde{\mathcal{O}}(1.4143^n)$. ❑

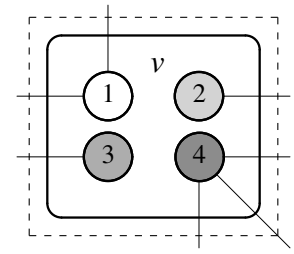
Übung 7.23. Konstruieren Sie eine $(3,2)$-CSP-Instanz mit drei Variablen so, dass sie nicht lösbar ist.

7.4.3 Ein deterministischer CSP-Algorithmus

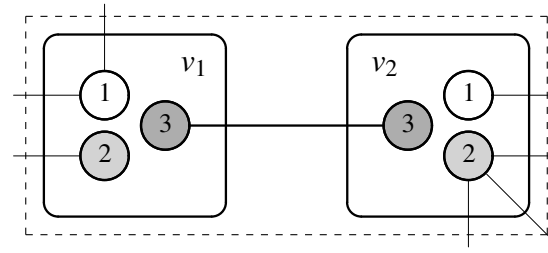
In diesem Abschnitt wenden wir uns dem angekündigten deterministischen Algorithmus für $(3,2)$-CSP von Beigel und Eppstein [BE05] zu, der eine Laufzeit von $\tilde{\mathcal{O}}(1.36443^n)$ hat (siehe Satz 7.50). Genau genommen kann dieser Algorithmus auch $(4,2)$-CSP lösen, da man eine gegebene $(4,2)$-CSP-Instanz wie folgt in eine äquivalente $(3,2)$-CSP-Instanz umformen kann (siehe auch Übung 7.24 unten):

1. Ersetze jedes Vorkommen einer Variablen v mit vier Farben durch zwei neue Variablen v_1 und v_2 mit je drei Farben, zwei von denen jeweils verschiedenen zwei der vier Farben von v entsprechen (z. B. entsprechen den Farben 1 und 3 (bzw. 2 und 4) von v in Abb. 7.17(a) die Farben 1 und 2 in v_1 (bzw. in v_2) in Abb. 7.17(b).
2. Füge eine neue Einschränkung hinzu, die die jeweils dritte Farbe der beiden neuen Variablen enthält (nämlich $((v_1,3),(v_2,3))$ in Abb. 7.17(b)).

Abbildung 7.17 zeigt die Transformation einer $(4,2)$-CSP-Instanz (Abb. 7.17(a)) in eine äquivalente $(3,2)$-CSP-Instanz (Abb. 7.17(b)). Offenbar kann diese Transformation auch umgekehrt ausgeführt werden, um eine gegebene $(3,2)$-CSP-Instanz in eine äquivalente$(4,2)$-CSP-Instanz umzuformen. Das heißt, eine „isolierte“ Einschränkung in einer $(3,2)$-CSP-Instanz, wie $((v_1,3),(v_2,3))$ in Abb. 7.17(b), kann entfernt und die verbleibenden Variablen v_1 und v_2 können zu einer Variablen v wie in Abb. 7.17(a) verschmolzen werden (siehe dazu auch Lemma 7.25).



(a) $(4,2)$-CSP-Instanz



(b) $(3,2)$-CSP-Instanz

Abb. 7.17. Transformation einer $(4,2)$-CSP-Instanz in eine äquivalente $(3,2)$-CSP-Instanz

Übung 7.24. Zeigen Sie, dass die in Abb. 7.17 dargestellte Transformation eine gegebene $(4,2)$-CSP-Instanz in eine *äquivalente* $(3,2)$-CSP-Instanz umformt.

Bezeichnet n_i die Anzahl der Variablen mit i Farben in einer $(4,2)$-CSP-Instanz I, so ist die Größe von I aufgrund dieser Transformation eigentlich $n = n_3 + 2n_4$.[41] Jedoch werden wir diese Größe stattdessen mit

$$n = n_3 + (2-\varepsilon)n_4 \tag{7.7}$$

ansetzen, wobei $\varepsilon \approx 0.095543$ eine später zu bestimmende Konstante ist. In dieser Weise können wir nämlich eine sehr nützliche Methode von Eppstein [Epp04] anwenden, mit der man mehrdimensionale Rekursionsgleichungen (die in diesem Fall von den Parametern n_3 und n_4 abhängen) auf eindimensionale Rekursionsgleichungen reduzieren kann, ohne das asymptotische Verhalten zu ändern.

Sei I also eine beliebige gegebene $(4,2)$-CSP-Instanz. Die Grundidee des Algorithmus liegt darin, lokale Situationen in I zu finden, die es erlauben, diese Instanz mittels konkreter Reduktionen durch wenige kleinere Instanzen zu ersetzen, von denen mindestens eine zu I äquivalent ist; dies garantiert, dass die Lösbarkeit der originalen Instanz I bei den Vereinfachungen erhalten bleibt. Auf diese kleineren Instanzen wird der Algorithmus rekursiv nach Art einer Backtracking-Strategie (siehe Abschnitt 3.5) angesetzt. Genauer gesagt soll I in jeder solchen Situation durch eine Reihe von Instanzen I_j der Größe $|I_j| = |I| - s_j$ mit $s_j > 0$ ersetzt werden, sodass I genau dann lösbar ist, wenn mindestens eine der Instanzen I_j lösbar ist. Tritt zum Beispiel *dieselbe* lokale Situation in jedem Reduktionsschritt des Algorithmus auf, so ergibt sich im schlimmsten Fall die Laufzeit des Algorithmus als Lösung einer Rekursion der Form

$$R(n) = \sum_j R(n-s_j) + p(n), \tag{7.8}$$

wobei p ein Polynom ist. Setzen wir $\alpha(s_1, s_2, \ldots)$ gleich der größten Nullstelle der charakteristischen Funktion

$$c(x) = 1 - \sum_j x^{-s_j},$$

so ist die Lösung von (7.8) in $\mathcal{O}(\alpha(s_1, s_2, \ldots)^n)$. Diesen Wert $\alpha(s_1, s_2, \ldots)$ nennen wir den *Arbeitsfaktor* der jeweils betrachteten lokalen Situation zur Vereinfachung der Instanz. Ist α der größte Arbeitsfaktor bezüglich der lokalen Situationen, die wir im Folgenden identifizieren werden, so können wir die Laufzeit des Algorithmus im schlimmsten Fall durch $\tilde{\mathcal{O}}(\alpha^n)$ abschätzen. Dabei machen wir von der oben erwähnten Eppstein-Methode Gebrauch, welche es erlaubt, mittels einer geeignet gewählten Konstante ε mehrdimensionale auf eindimensionale Rekursionsgleichungen zu reduzieren, ohne das asymptotische Verhalten zu ändern. Offenbar hängt α

Arbeitsfaktor

[41] Die Anzahlen n_1 und n_2 treten hier nicht auf, weil Variablen mit nur einer Farbe unmittelbar und Variablen mit nur zwei Farben nach Lemma 7.13.5 eliminiert werden können.

von ε ab. Insbesondere wird ε so gewählt werden, dass α möglichst klein ist. Initial gehen wir von einer Konstante ε mit $0 < \varepsilon < 1$ aus. In der nachfolgenden Analyse verschiedener lokaler Situationen wird ε noch enger eingeschränkt werden.

Die obigen Bemerkungen sind noch etwas vage und daher vielleicht noch nicht ganz verständlich. Dies wird aber klarer werden, wenn wir uns im Folgenden konkreten lokalen Situationen in $(4,2)$-CSP-Instanzen zuwenden und erklären, wie diese in vereinfachte Instanzen umgeformt werden können und welche Auswirkungen dies auf den jeweiligen Arbeitsfaktor hat. Dazu müssen wir eine Vielzahl verschiedener Fälle betrachten. Wer sich nicht so sehr dafür interessiert, wie die einzelnen Arbeitsfaktoren in diesen Fällen zustande kommen und begründet werden, kann die folgenden Seiten überspringen und gleich auf Seite 216 bei der Laufzeitanalyse des deterministischen CSP-Algorithmus von Beigel und Eppstein weiterlesen.

Isolierte Einschränkungen

isolierte Einschränkung

Wir kommen nun zur ersten lokalen Situation, die zur Vereinfachung einer gegebenen $(4,2)$-CSP-Instanz führen kann: *isolierte Einschränkungen*. Darunter verstehen wir Einschränkungen der Form $((u,i),(v,i))$, wobei u und v Variablen sind und i eine Farbe, sodass weder (u,i) noch (v,i) in irgendeiner anderen Einschränkung vorkommt. Kommt nur das Paar (u,i) in keiner anderen Einschränkung als dieser vor, das Paar (v,i) aber schon, dann nennen wir $((u,i),(v,i))$ eine *baumelnde Einschränkung*.

baumelnde Einschränkung

Ein Beispiel für eine isolierte Einschränkung ist $((v_1,3),(v_2,3))$ in Abb. 7.17(b). Wie wir mit solchen Einschränkungen umgehen können, zeigt das folgende Lemma. Im Folgenden gehen wir stets von einer reduzierten $(4,2)$-CSP-Instanz aus, einer Instanz also, auf die keine der Aussagen aus Lemma 7.13 mehr anwendbar ist.

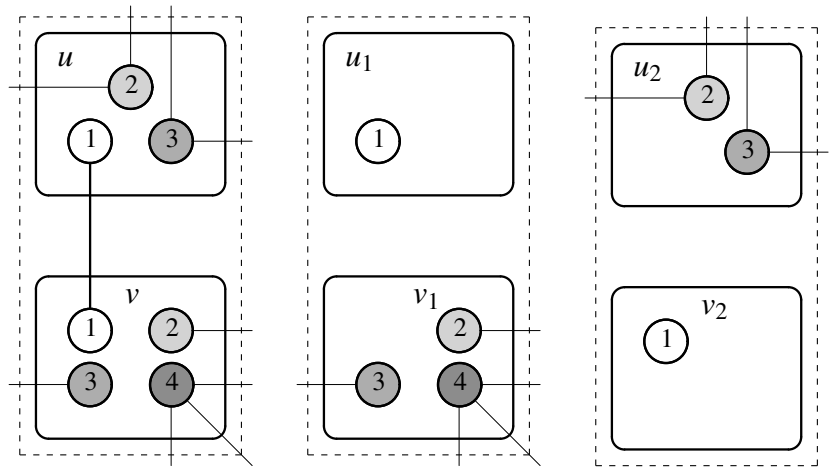
Lemma 7.25 (Beigel und Eppstein [BE05]). *Sei $((u,i),(v,i))$ eine isolierte Einschränkung in einer reduzierten $(4,2)$-CSP-Instanz I und sei $\varepsilon \leq 0.545$. Dann kann I durch kleinere Instanzen mit Arbeitsfaktor höchstens $\alpha(2-\varepsilon,3-\varepsilon)$ ersetzt werden, sodass I genau dann lösbar ist, wenn mindestens eine dieser kleineren Instanzen lösbar ist.*

Beweis. Angenommen, den beiden Variablen u und v stehen jeweils drei Farben zur Verfügung. Dann können wir gemäß der in Abb. 7.17 dargestellten Transformation die Variablen u und v in I durch eine neue Variable w mit vier Farben ersetzen, sodass die resultierende Instanz I' zu I äquivalent ist. Da wir dabei zwei 3-Farben-Variablen durch eine 4-Farben-Variable ersetzen, folgt aus (7.7), dass sich die Größe n' der Instanz I', nämlich

$$n' = (n_3 - 2) + (2-\varepsilon)(n_4+1) = n_3 + (2-\varepsilon)n_4 - \varepsilon = n - \varepsilon,$$

gegenüber der Größe $n = n_3 + (2-\varepsilon)n_4$ von I bereits um ε verringert hat, wobei n_3 bzw. n_4 die Anzahl der Variablen in I mit 3 bzw. 4 Farben bezeichnet.

Nehmen wir nun an, dass entweder u oder v in I vier Farben zur Verfügung stehen. Wenn es in diesem Fall überhaupt eine Färbung gibt, die alle Einschränkungen



(a) $(4,2)$-CSP-Instanz I (b) Transformierte Instanz I_1 (c) Transformierte Instanz I_2

Abb. 7.18. Transformation einer isolierten Einschränkung in einer $(4,2)$-CSP-Instanz

erfüllt, dann gibt es auch eine Färbung, in der genau eine der Variablen u und v mit 1 gefärbt wird. Nehmen wir zunächst an, dass v vier Farben zur Verfügung stehen, u aber nur drei. Wie in Abb. 7.18 dargestellt, können wir I in eine von zwei Instanzen umformen, I_1 und I_2, sodass I genau dann lösbar ist, wenn I_1 oder I_2 lösbar ist. Wir betrachten dabei zwei Fälle:

1. Ist insbesondere I durch eine Färbung lösbar, in der die Variable u mit 1 gefärbt wird, so gibt es eine Färbung, die I_1 löst. Deshalb transformieren wir I in I_1.
2. Ist I hingegen durch eine Färbung lösbar, in der die Variable v mit 1 gefärbt wird, so gibt es eine Färbung, die I_2 löst. Deshalb transformieren wir I in I_2.

Im ersten Fall reduziert sich die Größe des Problems um $2-\varepsilon$, denn die Variable u_1 kann unmittelbar von I_1 entfernt werden, während v_1 zwar eine Farbe weniger als v hat, aber nicht entfernt wird. Es ergibt sich also für die Größe n' von I_1:

$$n' = (n_3 - 1 + 1) + (2-\varepsilon)(n_4 - 1) = n_3 + (2-\varepsilon)n_4 - (2-\varepsilon) = n - (2-\varepsilon).$$

Im zweiten Fall reduziert sich die Größe des Problems sogar um $3-\varepsilon$, denn die Variable v_2 kann unmittelbar von I_2 entfernt werden, während u_2 nur noch zwei Farben hat und deshalb nach der letzten Aussage von Lemma 7.13 entfernt werden kann. Es ergibt sich also für die Größe n'' von I_2:

$$n'' = (n_3 - 1) + (2-\varepsilon)(n_4 - 1) = n_3 + (2-\varepsilon)n_4 - 1 - (2-\varepsilon) = n - (3-\varepsilon).$$

Daraus folgt ein Arbeitsfaktor von $\alpha(2-\varepsilon, 3-\varepsilon)$. Denselben Arbeitsfaktor erhalten wir natürlich im symmetrischen Fall, dass u vier Farben, v aber nur drei Farben zur Verfügung stehen.

Nehmen wir schließlich an, dass sowohl u als auch v vier Farben zur Verfügung stehen, so ergibt sich ein Arbeitsfaktor von $\alpha(3-2\varepsilon,3-2\varepsilon)$ (siehe Übung 7.26). Da dieser jedoch für $\varepsilon \leq 0.545$ kleiner als $\alpha(2-\varepsilon,3-\varepsilon)$ ist, ist nur der letzere relevant. ❑

Übung 7.26. Betrachten Sie im Beweis von Lemma 7.25 den Fall, dass sowohl u als auch v vier Farben zur Verfügung stehen. Zeigen Sie, dass sich dann ein Arbeitsfaktor von

$$\alpha(3-2\varepsilon,3-2\varepsilon)$$

ergibt.

Baumelnde Einschränkungen

In lokalen Situationen mit baumelnden Einschränkungen, wie sie im Absatz vor Lemma 7.25 definiert wurden, erhalten wir denselben Arbeitsfaktor wie für isolierte Einschränkungen.

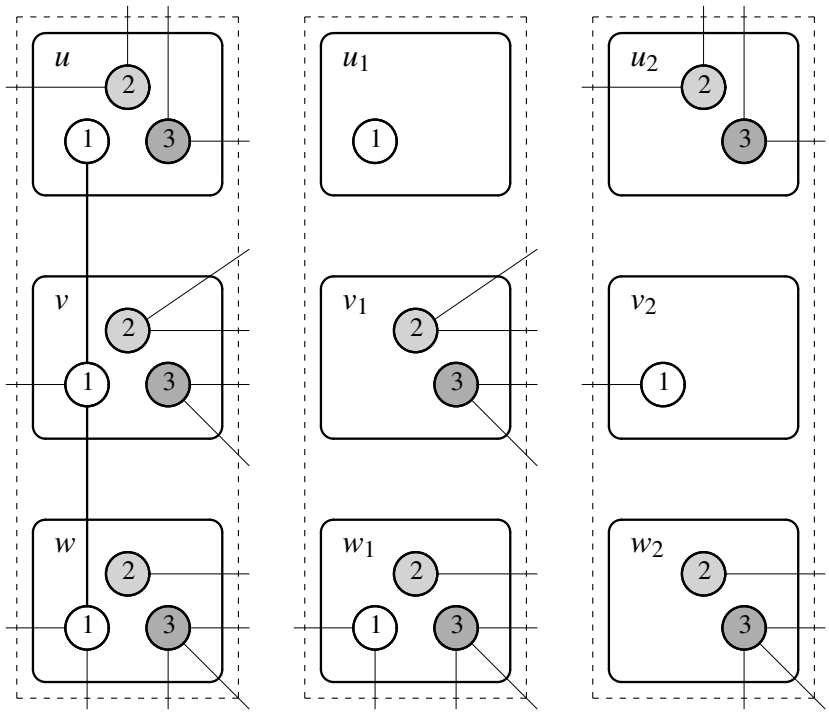
Lemma 7.27 (Beigel und Eppstein [BE05]). *Sei $((u,i),(v,i))$ eine baumelnde Einschränkung in einer reduzierten $(4,2)$-CSP-Instanz I. Dann kann I durch kleinere Instanzen mit Arbeitsfaktor höchstens $\alpha(2-\varepsilon,3-\varepsilon)$ ersetzt werden, sodass I genau dann lösbar ist, wenn mindestens eine dieser kleineren Instanzen lösbar ist.*

Beweis. Durch Umbenennen der Farben (falls nötig) können wir sicherstellen, dass unsere baumelnde Einschränkung $((u,i),(v,i))$ in I die Farbe $i=1$ enthält. Nach Definition kommt das Paar $(u,1)$ also in keiner anderen Einschränkung von I vor, aber das Paar $(v,1)$ erscheint noch in mindestens einer anderen Einschränkung, etwa in $((v,1),(w,j))$. Offenbar muss dabei $w \neq u$ gelten, denn sonst könnten wir Lemma 7.13.3 anwenden; I ist jedoch schon reduziert. Ein Beispiel ist in Abb. 7.19(a) zu sehen.

Wie in Abb. 7.19 dargestellt, können wir I in eine von zwei Instanzen umformen, I_1 und I_2, sodass I genau dann lösbar ist, wenn I_1 oder I_2 lösbar ist. Wir betrachten dabei die folgenden beiden Fälle:

1. Ist insbesondere I durch eine Färbung lösbar, in der die Variable v nicht mit 1 gefärbt wird, so gibt es eine Färbung der Variablen von I_1 (siehe Abb. 7.19(b)), die u_1 mit Farbe 1 färbt und I_1 löst. Deshalb transformieren wir I in I_1.
2. Ist I hingegen durch eine Färbung lösbar, in der die Variable v mit 1 gefärbt wird, so gibt es eine Färbung der Variablen von I_2 (siehe Abb. 7.19(c)), die die Farbe 1 für u_2 und die Farbe j für w_2 (in Abb. 7.19(c) ist $j=1$) ausschließt und I_2 löst. Deshalb transformieren wir I in I_2.

Um den größtmöglichen Arbeitsfaktor, der sich dabei ergibt, ermitteln zu können, stellen wir zunächst fest, dass die Variable w aus der Einschränkung $((v,1),(w,j))$ im schlimmsten Fall vier Farben haben kann. Entfernen wir eine davon, so reduziert sich die Problemgröße um lediglich $1-\varepsilon$. Da I reduziert ist, haben alle Variablen in I mindestens drei Farben. In Abhängigkeit davon, wie viele Farben u und v haben, erhalten wir unterschiedliche Arbeitsfaktoren in den folgenden vier Fällen:



(a) $(4,2)$-CSP-Instanz I (b) Transformierte Instanz I_1 (c) Transformierte Instanz I_2

Abb. 7.19. Transformation einer baumelnden Einschränkung in einer $(4,2)$-CSP-Instanz

1. Haben sowohl u als auch v drei Farben (wie in Abb. 7.19(a)), so ergibt sich ein Arbeitsfaktor von
$$\alpha(2,3-\varepsilon). \tag{7.9}$$
2. Hat nur u vier Farben, v aber drei, so ist der Arbeitsfaktor
$$\alpha(3-\varepsilon,3-2\varepsilon). \tag{7.10}$$
3. Hat nur v vier Farben, u aber drei, so ist der Arbeitsfaktor
$$\alpha(2-\varepsilon,4-2\varepsilon). \tag{7.11}$$
4. Haben sowohl u als auch v vier Farben, so ergibt sich ein Arbeitsfaktor von
$$\alpha(3-2\varepsilon,4-3\varepsilon). \tag{7.12}$$

Wir zeigen (7.9) im ersten Fall (u und v haben beide drei Farben) anhand von Abb. 7.19. Entscheiden wir uns nämlich dafür, I in I_1 zu transformieren (weil v in I nicht mit der Farbe 1 gefärbt wird, siehe Abb. 7.19(b)), so kann die einfarbige Variable u_1 unmittelbar entfernt werden, während die Variable v_1 nur noch zwei Farben

hat und deshalb nach der letzten Aussage von Lemma 7.13 entfernt werden kann. In diesem Fall hat die Variable w_1 in I_1 ebenso viele Farben wie die ursprüngliche Variable w in I. Ist $n = n_3 + (2-\varepsilon)n_4$ also die Größe der Instanz I, wobei n_3 bzw. n_4 wieder die Anzahl der Variablen in I mit 3 bzw. 4 Farben bezeichnet, so ergibt sich für die Größe n' von I_1:

$$n' = (n_3 - 2) + (2-\varepsilon)n_4 = n - 2.$$

Entscheiden wir uns andererseits dafür, I in I_2 zu transformieren (weil v in I mit der Farbe 1 gefärbt wird, siehe Abb. 7.19(c)), so kann die einfarbige Variable v_2 unmittelbar entfernt werden, während die Variable u_2 nur noch zwei Farben hat und deshalb wieder nach der letzten Aussage von Lemma 7.13 entfernt werden kann. Die Variable w_2 schließlich trägt zur Verminderung der Problemgröße im schlimmsten Fall nur den Betrag $1-\varepsilon$ bei, da eine von möglicherweise vier Farben entfernt wird. Insgesamt ergibt sich somit für die Größe n'' von I_2 eine obere Schranke von

$$n'' = (n_3 - 2 + 1) + (2-\varepsilon)(n_4 - 1) = n_3 + (2-\varepsilon)n_4 - (3-\varepsilon) = n - (3-\varepsilon),$$

womit der Arbeitsfaktor von $\alpha(2, 3-\varepsilon)$ aus (7.9) gezeigt ist. Die in (7.10) bis (7.12) angegebenen Arbeitsfaktoren lassen sich in ähnlicher Weise bestimmen (siehe Übung 7.28). Alle vier Arbeitsfaktoren in (7.9) bis (7.12) werden durch

$$\alpha(2-\varepsilon, 3-\varepsilon)$$

dominiert. ❑

Übung 7.28. Bestimmen Sie im Beweis von Lemma 7.27 die in (7.10) bis (7.12) angegebenen Arbeitsfaktoren.

Implikationen zwischen Färbungen

Nun betrachten wir lokale Situationen einer gegebenen $(4,2)$-CSP-Instanz, in denen mehrere Einschränkungen zwischen zwei Variablen dasselbe Paar (v,i) enthalten, beispielsweise zwei Einschränkungen der Form $((v,i),(w,j))$ und $((v,i),(w,k))$, wobei $|\{i,j,k\}| = 3$. Man kann sich (v,i) bildlich wie ein „Scharnier“ vorstellen, das mit verschiedenen Farben einer anderen Variablen verbunden ist. Im Extremfall, dass (v,i) mit jeder von i verschiedenen Farbe der Variablen w durch eine Einschränkung verbunden ist, sagen wir, (v,i) *impliziert* (w,i) (kurz mit $(v,i) \Longrightarrow (w,i)$ bezeichnet). Denn färbt man die Variable v mit i, so muss man auch w mit i färben, will man alle diese Einschränkungen erfüllen.

Implikation $(v,i) \Longrightarrow (w,i)$

Eine Bemerkung noch zum Unterschied der oben beschriebenen Eigenschaft von Einschränkungen und der in Lemma 7.13.3 beschriebenen Eigenschaft. Werfen wir zum Beispiel einen Blick auf die Variablen Eva, Lisa und Lotte in Abb. 7.6 und auf die Einschränkungen $((\text{Eva},1),(\text{Lotte},1))$ und $((\text{Eva},1),(\text{Lotte},3))$, dann sieht das Paar $(\text{Eva},1)$ ebenfalls „scharnierartig“ aus. Da es aber auch Einschränkungen

$((\text{Lisa},3),(\text{Lotte},1))$ und $((\text{Lisa},3),(\text{Lotte},3))$ gibt, kann eine der Farben 1 und 3 aus der Variablen Lotte gemäß Lemma 7.13.3 entfernt werden (siehe Übung 7.29). In einer reduzierten $(3,2)$-CSP- oder $(4,2)$-CSP-Instanz I kommen solche Fälle also nicht mehr vor. Jedoch kann es in I noch Einschränkungen der Form $((v,i),(w,j))$ und $((v,i),(w,k))$ geben, bei denen die Paare (w,j) und (w,k) nicht nur mit (v,i), sondern auch mit *verschiedenen* anderen Paaren durch Einschränkungen verbunden sind, sodass Lemma 7.13.3 nicht anwendbar ist. Wie wir mit solchen Fällen umgehen können, sagt uns das folgende Lemma.

Übung 7.29. Wenden Sie Lemma 7.13.3 auf die $(3,2)$-CSP-Instanz in Abb. 7.6 an.

Lemma 7.30 (Beigel und Eppstein [BE05]). *Sei I eine reduzierte $(4,2)$-CSP-Instanz, die zwei Einschränkungen der Form $((v,i),(w,j))$ und $((v,i),(w,k))$ enthält, wobei $|\{i,j,k\}|=3$. Weiterhin sei $\varepsilon \leq 0.4$. Dann kann I durch kleinere Instanzen mit Arbeitsfaktor höchstens $\alpha(2-\varepsilon,3-2\varepsilon)$ ersetzt werden, sodass I genau dann lösbar ist, wenn mindestens eine dieser kleineren Instanzen lösbar ist.*

Beweis. Zunächst dürfen wir annehmen, dass in I keine Farbe einer Variablen nur in einer einzigen Einschränkung vorkommt, denn eine solche Einschränkung wäre isoliert oder baumelnd und könnte mit Lemma 7.25 oder 7.27 entfernt werden.

Wir unterscheiden drei Fälle:

1. Es gibt in I zwei Einschränkungen der Form $((v,i),(w,j))$ und $((v,i),(w,k))$, mit $|\{i,j,k\}|=3$, die aber keine Implikation der Form $(v,i) \Longrightarrow (w,i)$ bilden.
2. Es gibt eine Implikation $(u,i) \Longrightarrow (v,i)$ in I, sodass es keine Implikation der Form $(v,i) \Longrightarrow (w,i)$ in I gibt.
3. Für jede Implikation $(u,i) \Longrightarrow (v,i)$ in I gibt es in I eine Implikation der Form $(v,i) \Longrightarrow (w,i)$.

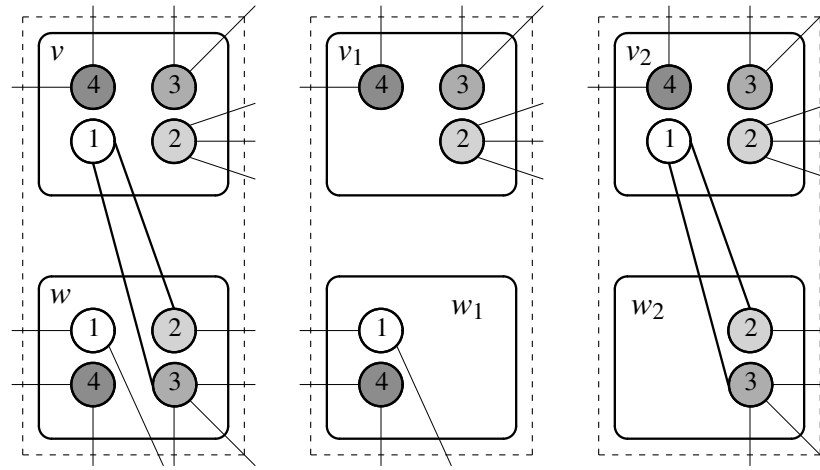
Im ersten Fall müssen für w vier Farben zur Verfügung stehen, von denen genau zwei mit dem Paar (v,i) eine Einschränkung bilden. Falls nötig, benennen wir die Farben so um, dass $((v,1),(w,2))$ und $((v,1),(w,3))$ diese beiden Einschränkungen sind (siehe Abb. 7.20(a)).

Schränken wir w auf die Farben 1 und 4 ein, so transformieren wir I in I_1 und erhalten in I_1 die 2-Farben-Variable w_1 (siehe Abb. 7.20(b)), die wir nach der letzten Aussage von Lemma 7.13 entfernen können. Da dabei die 4-Farben-Variable w verschwindet und sich die 4-Farben-Variable v in eine 3-Farben-Variable v_1 verwandelt, ergibt sich aus der Größe $n = n_3 + (2-\varepsilon)n_4$ für I die Größe

$$n' = (n_3+1) + (2-\varepsilon)(n_4-2) = n - (3-2\varepsilon)$$

von I_1, wobei n_3 bzw. n_4 wieder die Anzahl der Variablen mit 3 bzw. 4 Farben bezeichnet.

Schränken wir w jedoch auf die Farben 2 und 3 ein, so transformieren wir I in I_2 und erhalten in I_2 die 2-Farben-Variable w_2 (siehe Abb. 7.20(c)), die wir ebenso nach der letzten Aussage von Lemma 7.13 entfernen können. Hier werden die Farben, die v_2 zur Verfügung stehen, allerdings nicht verringert. Es ergibt sich also die Größe



(a) $(4,2)$-CSP-Instanz I (b) Transformierte Instanz I_1 (c) Transformierte Instanz I_2

Abb. 7.20. Erste Transformation einer $(4,2)$-CSP-Instanz im Beweis von Lemma 7.30

$$n'' = n_3 + (2-\varepsilon)(n_4 - 1) = n - (2-\varepsilon)$$

von I_2 und somit der Arbeitsfaktor $\alpha(2-\varepsilon, 3-2\varepsilon)$ im ersten Fall.

Im zweiten Fall betrachten wir Implikationen der Form $(u,i) \Longrightarrow (v,i)$ in I, für die es keine Implikation der Form $(v,i) \Longrightarrow (w,i)$ in I gibt. Abbildung 7.21(a) zeigt ein Beispiel.

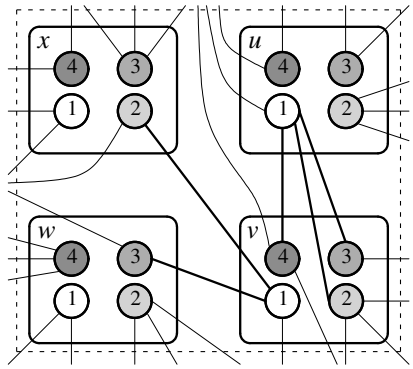
Schließen wir die Farbe i für die Variable v aus, so zwingen uns die Einschränkungen zwischen u und v, auch für u die Farbe i auszuschließen, denn wegen der Implikation $(u,i) \Longrightarrow (v,i)$ in I ist jedes Paar (v,j), $j \neq i$, mit (u,i) verbunden. Somit erhalten wir die um eine Farbe reduzierten Variablen u_1 und v_1 in der neuen Instanz I_1 (siehe Abb. 7.21(b)). Färben wir jedoch v mit i (schließen also für v alle Farben $j \neq i$ aus), so kann diese Variable entfernt werden und auch je eine Farbe in mindestens zwei anderen Variablen, und wir erhalten die neue Instanz I_2 (siehe Abb. 7.21(c)). Offenbar ist I genau dann erfüllbar, wenn I_1 oder I_2 erfüllbar ist.

Schlimmstenfalls kann die Variable u vier Farben haben. Der Arbeitsfaktor ergibt sich nun abhängig davon, wie viele Farben der Variablen v in I zur Verfügung stehen. Sind es – wie in Abb. 7.21(a) – ebenfalls vier Farben, so wandelt der Ausschluss der Farbe i für v die beiden 4-Farben-Variablen u und v in I in die 3-Farben-Variablen u_1 und v_1 in I_1 um, und I_1 hat somit die Größe

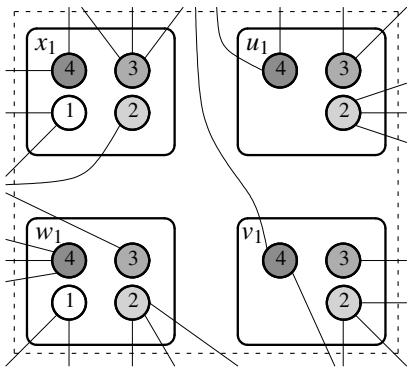
$$n' = (n_3 + 2) + (2-\varepsilon)(n_4 - 2) = n_3 + (2-\varepsilon)n_4 - (2-2\varepsilon) = n - (2-2\varepsilon).$$

Das Färben von v mit i hingegen eliminiert diese Variable und mindestens zwei Farben in anderen Variablen (die auch vier Farben haben können), weshalb in diesem Fall I_2 die Größe

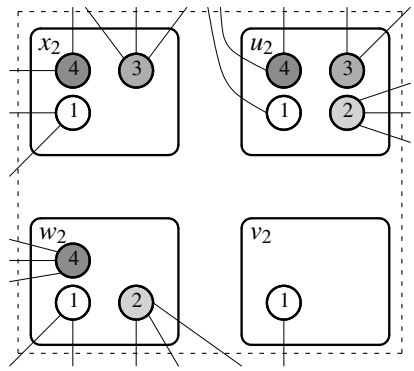
$$n'' = (n_3 + 2) + (2-\varepsilon)(n_4 - 3) = n_3 + (2-\varepsilon)n_4 - (4-3\varepsilon) = n - (4-3\varepsilon)$$



(a) $(4,2)$-CSP-Instanz I



(b) Transformierte Instanz I_1



(c) Transformierte Instanz I_2

Abb. 7.21. Zweite Transformation einer $(4,2)$-CSP-Instanz im Beweis von Lemma 7.30

hat. Also ist der Arbeitsfaktor in diesem Fall $\alpha(2-2\varepsilon,4-3\varepsilon)$.

Kann v jedoch nur aus drei Farben wählen, so überführt der Ausschluss von i für v die 4-Farben-Variable u in die 3-Farben-Variable u_1 in I_1 und die 3-Farben-Variable v in die 2-Farben-Variable v_1, die mit der letzten Aussage von Lemma 7.13 entfernt werden kann. Es ergibt sich die Größe

$$n' = (n_3+1-1)+(2-\varepsilon)(n_4-1) = n_3+(2-\varepsilon)n_4-(2-\varepsilon) = n-(2-\varepsilon)$$

von I_1. Das Färben von v mit i führt in diesem Fall dazu, dass die 3-Farben-Variable v verschwindet und ebenso zwei Farben aus anderen Variablen, wodurch zwei 4-Farben-Variablen in zwei 3-Farben-Variablen umgeformt werden können. Somit ist in diesem Fall

$$n'' = (n_3-1+2)+(2-\varepsilon)(n_4-2) = n_3+(2-\varepsilon)n_4-(3-2\varepsilon) = n-(3-2\varepsilon)$$

die Größe von I_2. Wir erhalten also einen Arbeitsfaktor von $\alpha(2-\varepsilon,3-2\varepsilon)$.

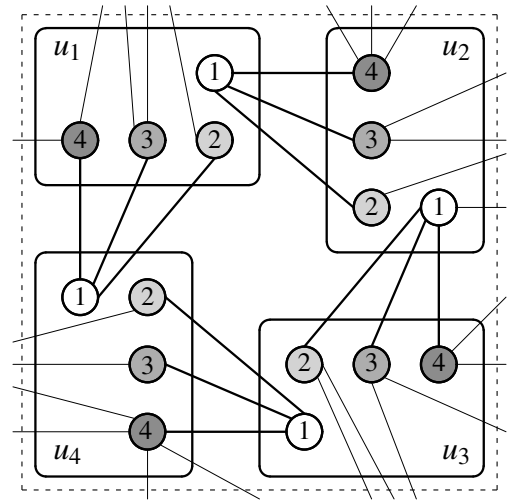


Abb. 7.22. Ein Zyklus von Implikationen in der $(4,2)$-CSP-Instanz I für Lemma 7.30

Der dritte Fall schließlich sagt, dass es für jede Implikation der Form $(u,i) \Longrightarrow (v,i)$ in I eine Implikation der Form $(v,i) \Longrightarrow (w,i)$ gibt. Dann können wir einen Zyklus von Paaren finden, sodass ein Paar das andere impliziert:

$$(u_1,i) \Longrightarrow (u_2,i),\ (u_2,i) \Longrightarrow (u_3,i),\ \ldots,(u_{k-1},i) \Longrightarrow (u_k,i),\ (u_k,i) \Longrightarrow (u_1,i).$$

Abbildung 7.22 gibt ein Beispiel für einen Zyklus mit den vier Variablen u_1, u_2, u_3 und u_4 und der Farbe $i = 1$ an. Ist dabei keines der Paare (u_j,i), $1 \leq j \leq k$, an einer Einschränkung mit anderen Variablen beteiligt, so können wir jedes u_j mit i färben und werden den gesamten Zyklus auf einen Schlag los, d. h., die Problemgröße verringert sich in diesem Fall um k Variablen. Wird jedoch eines dieser Paare (u_ℓ,i) (wie das Paar $(u_2,1)$ im Beispiel von Abb. 7.22) durch ein außerhalb des Zyklus liegendes Paar (v,h) eingeschränkt, so muss

1. jede Lösung, die u_ℓ mit i färbt, auch alle übrigen Variablen im Zyklus mit i färben, und dann kann (v,h) ausgeschlossen werden;
2. jede Lösung, die i für u_ℓ verbietet, die Farbe i auch für alle übrigen Variablen im Zyklus verbieten.

Der größte Arbeitsfaktor ist in diesem Fall $\alpha(2,3-\varepsilon)$, und er tritt ein, wenn der Zyklus aus genau zwei Variablen besteht, denen jeweils nur drei Farben zur Verfügung stehen.

Alle vier Arbeitsfaktoren in den oben genannten drei Fällen werden für $\varepsilon \leq 0.4$ durch

$$\alpha(2-\varepsilon,3-2\varepsilon)$$

dominiert. ❑

Stark eingeschränkte Farben

Nun beschäftigen wir uns mit lokalen Situationen, in denen die Wahl einer Farbe für eine Variable viele andere Möglichkeiten, Variablen zu färben, ausschließt.

Lemma 7.31 (Beigel und Eppstein [BE05]). *Sei I eine reduzierte $(4,2)$-CSP-Instanz, ein Paar (v,i) enthält, sodass entweder*

- *(v,i) zu drei oder mehr Einschränkungen gehört und v vier Farben verfügbar sind oder*
- *(v,i) zu vier oder mehr Einschränkungen gehört und v drei Farben verfügbar sind.*

Dann kann I durch kleinere Instanzen mit Arbeitsfaktor höchstens $\alpha(1-\varepsilon, 5-4\varepsilon)$ ersetzt werden, sodass I genau dann lösbar ist, wenn mindestens eine dieser kleineren Instanzen lösbar ist. **ohne Beweis**

Übung 7.32. Beweisen Sie Lemma 7.31. **Hinweis:** Sie dürfen dabei annehmen, dass jede Einschränkung, in der das Paar (v,i) aus Lemma 7.31 vorkommt, dieses Paar mit jeweils *verschiedenen* anderen Variablen verbindet. Das heißt, Einschränkungen der Form $((v,i),(w,j))$ und $((v,i),(w,k))$ müssen Sie nicht betrachten, denn diese können bereits durch Lemma 7.30 behandelt werden.

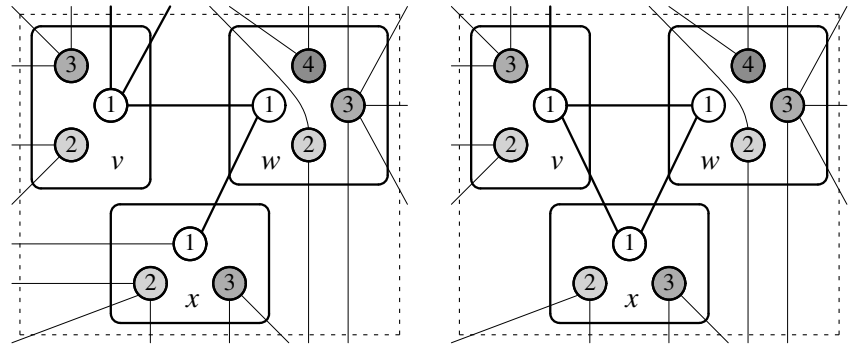
Die folgenden beiden Lemmata zeigen, wie solche Instanzen vereinfacht werden können, in denen die zwei Paare einer Einschränkung an verschieden vielen Einschränkungen beteiligt sind.

Lemma 7.33 (Beigel und Eppstein [BE05]). *Sei I eine reduzierte $(4,2)$-CSP-Instanz, auf die keines der Lemmata 7.25, 7.27, 7.30 und 7.31 anwendbar ist und die ein Paar (v,i) enthält, das in drei Einschränkungen vorkommt, und eine dieser Einschränkungen verbindet es mit einer 4-Farben-Variablen w. Weiterhin sei $\varepsilon \leq 0.3576$. Dann kann I durch kleinere Instanzen mit Arbeitsfaktor höchstens $\alpha(3-\varepsilon, 4-\varepsilon, 4-\varepsilon)$ ersetzt werden, sodass I genau dann lösbar ist, wenn mindestens eine dieser kleineren Instanzen lösbar ist.*

Beweis. Der Einfachheit halber nehmen wir an, dass $((v,1),(w,1))$ die im Lemma erwähnte Einschränkung ist. Wie in Abb. 7.23 dargestellt unterscheiden wir zwei Fälle:

1. $(v,1)$ und $(w,1)$ bilden kein Dreieck mit einem dritten Paar $(x,1)$ (Abb. 7.23(a));
2. $(v,1)$ und $(w,1)$ bilden ein Dreieck mit einem dritten Paar $(x,1)$ (Abb. 7.23(b)).

Im ersten Fall gibt es also die Einschränkungen $((v,1),(w,1))$ und $((w,1),(x,1))$, aber nicht die Einschränkung $((v,1),(x,1))$ (siehe Abb. 7.23(a)). Da Lemma 7.31 nicht anwendbar ist, kann v nur drei Farben zur Auswahl haben. Wir können entweder v mit 1 färben oder aber uns entscheiden, dies nicht zu tun. Färben wir v mit 1, so können wir die Variable v nach der letzten Aussage von Lemma 7.13 entfernen und auch die drei Farben in anderen Variablen, mit denen v durch die drei



(a) $(v,1)$, $(w,1)$ und $(x,1)$ bilden kein Dreieck (b) $(v,1)$, $(w,1)$ und $(x,1)$ bilden ein Dreieck

Abb. 7.23. Zwei Fälle im Beweis von Lemma 7.33

Einschränkungen verbunden ist. Schließen wir die Färbung $(v,1)$ jedoch aus, so entsteht – zusätzlich zur Eliminierung von v – eine baumelnde Einschränkung bei $(w,1)$, die wir nach Lemma 7.27 für eine weitere Unterteilung der Instanz in kleinere Instanzen nutzen können. Da w vier Farben zur Verfügung stehen, ergibt sich aus dem Beweis von Lemma 7.27 dabei ein Arbeitsfaktor von $\alpha(3-\varepsilon,3-2\varepsilon)$, sodass in diesem Fall insgesamt ein Arbeitsfaktor von $\alpha(4-\varepsilon,4-2\varepsilon,4-3\varepsilon)$ zu Buche schlägt (siehe Übung 7.34 unten).

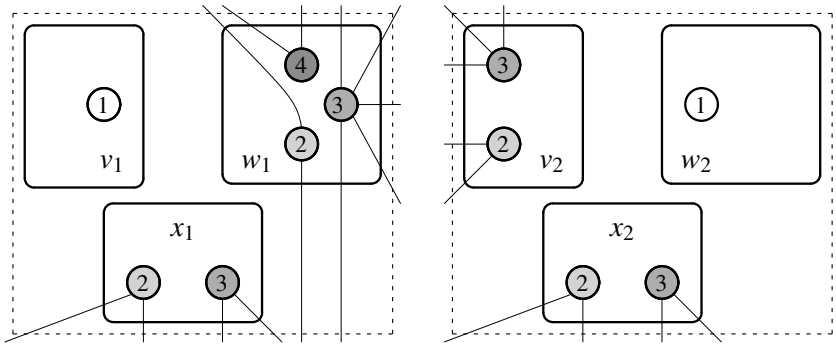
Wie in Abb. 7.23(b) dargestellt, gibt es im zweiten Fall die Einschränkungen $((v,1),(w,1))$, $((w,1),(x,1))$ und $((v,1),(x,1))$. Da w eine 4-Farben-Variable ist, Lemma 7.31 aber nicht anwendbar ist, kann $(w,1)$ nur an zwei Einschränkungen beteiligt sein, nämlich einer mit $(v,1)$ und einer mit $(x,1)$. Werden beide Färbungen, $(v,1)$ und $(x,1)$, ausgeschlossen, können wir also bedenkenlos w mit 1 färben. Daher unterteilen wir die gegebene Instanz I in drei kleinere Instanzen, I_1, I_2 und I_3, je eine für die Färbungen $(v,1)$, $(w,1)$ und $(x,1)$ (siehe Abb. 7.24). Offenbar ist mindestens eine der kleineren Instanzen äquivalent zu I.

Zunächst nehmen wir an, dass x eine Variable ist, der drei Farben zur Verfügung stehen. Der schlechteste Fall tritt ein, wenn $(x,1)$ nur an zwei Einschränkungen beteiligt ist, nämlich der mit $(v,1)$ und der mit $(w,1)$. Wird nun die Färbung $(v,1)$ gewählt, so werden in I_1 die mit $(v,1)$ verbundenen Farben ausgeschlossen (siehe Abb. 7.24(a)). Dann hat w_1 drei Farben, während v_1, x_1 sowie die dritte Nachbarvariable[42] von $(v,1)$ in I_1 ganz eliminiert werden können. Die Größe von I_1 reduziert sich gegenüber der Größe $n = n_3 + (2-\varepsilon)n_4$ von I demnach auf

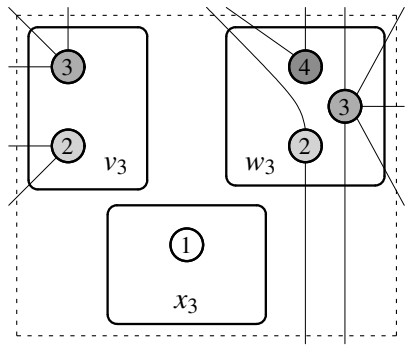
$$n' = (n_3 - 3 + 1) + (2-\varepsilon)(n_4 - 1) = n - (4-\varepsilon),$$

wobei n_3 bzw. n_4 wieder die Anzahl der Variablen in I mit 3 bzw. 4 Farben bezeichnen. Dieselbe Reduktion der Größe um $4-\varepsilon$ erhalten wir für I_2, wenn $(w,1)$ gewählt wird (siehe Abb. 7.24(b)), wodurch v_2, w_2 und x_2 eliminiert werden. Wird dagegen

[42] Dieser stehen dann wie x_1 nur noch zwei Farben zur Auswahl, denn hätte sie ursprünglich in I vier Farben gehabt, könnte sie wie im ersten Fall dieses Beweises behandelt werden.



(a) $(v,1)$ wird gewählt (b) $(w,1)$ wird gewählt



(c) $(x,1)$ wird gewählt

Abb. 7.24. Reduzierte Instanzen im zweiten Fall im Beweis von Lemma 7.33

$(x,1)$ gewählt (siehe Abb. 7.24(c)), so können wir v_3 und x_3 in I_3 eliminieren und w_3 hat drei Farben. Wir erhalten also für I_3 die Größe

$$n'' = (n_3 - 2 + 1) + (2 - \varepsilon)(n_4 - 1) = n - (3 - \varepsilon).$$

Für eine 3-Farben-Variable x in I ergibt sich also der Arbeitsfaktor $\alpha(3-\varepsilon, 4-\varepsilon, 4-\varepsilon)$.

Ist x eine Variable mit vier Farben, so ergibt sich entsprechend der Arbeitsfaktor $\alpha(4-2\varepsilon, 4-2\varepsilon, 4-2\varepsilon)$. Für $\varepsilon \leq 0.3576$ wird dieser jedoch von $\alpha(3-\varepsilon, 4-\varepsilon, 4-\varepsilon)$ dominiert. ❑

Übung 7.34. Argumentieren Sie im Detail, wie sich der Arbeitsfaktor

$$\alpha(4-\varepsilon, 4-2\varepsilon, 4-3\varepsilon)$$

im Beweis von Lemma 7.33 begründen lässt, falls (v,i) und (w,i) kein Dreieck mit einem dritten Paar (x,i) (Abb. 7.23(a)) bilden.

Lemma 7.35 (Beigel und Eppstein [BE05]). *Sei I eine reduzierte $(4,2)$-CSP-Instanz, auf die keines der Lemmata 7.25, 7.27, 7.30, 7.31 und 7.33 anwendbar ist und die ein Paar (v,i) enthält, das in drei Einschränkungen vorkommt, und eine dieser Einschränkungen verbindet es mit einem Paar (w,j), das in zwei Einschränkungen vorkommt. Dann kann I durch kleinere Instanzen mit Arbeitsfaktor höchstens $\max\{\alpha(1+\varepsilon,4),\alpha(3,4-\varepsilon,4)\}$ ersetzt werden, sodass I genau dann lösbar ist, wenn mindestens eine dieser kleineren Instanzen lösbar ist.*

Beweis. Der Einfachheit halber nehmen wir $j=i=1$ an, sodass $((v,1),(w,1))$ die im Lemma erwähnte Einschränkung ist. Da keines der früheren Lemmata anwendbar ist, stehen allen Nachbarvariablen von $(v,1)$ – und insbesondere w – drei Farben zur Verfügung. Wie in Abb. 7.25 dargestellt unterscheiden wir drei Fälle:

1. $(v,1)$ und $(w,1)$ bilden kein Dreieck mit einem dritten Paar $(x,1)$ (Abb. 7.25(a));
2. $(v,1)$ und $(w,1)$ bilden ein Dreieck mit einem dritten Paar $(x,1)$, das in drei Einschränkungen vorkommt (Abb. 7.25(b)).
3. $(v,1)$ und $(w,1)$ bilden ein Dreieck mit einem dritten Paar $(x,1)$, das in zwei Einschränkungen vorkommt (Abb. 7.25(c)).

Im ersten Fall gibt es zwar die Einschränkungen $((v,1),(w,1))$ und $((w,1),(x,1))$, aber nicht die Einschränkung $((v,1),(x,1))$ (siehe Abb. 7.25(a)). Färben wir die Variable v mit der Farbe 1, so eliminieren wir vier 3-Farben-Variablen: v und die drei Nachbarvariablen von v, die mit $(v,1)$ eine Einschränkung teilen. Schließen wir dagegen die Farbe 1 für v aus, so entsteht – zusätzlich zur Eliminierung von v – eine baumelnde Einschränkung bei $(w,1)$, die wir nach Lemma 7.27 für eine weitere Unterteilung der Instanz in kleinere Instanzen nutzen können. Da sowohl w als auch x drei Farben zur Verfügung stehen, ergibt sich aus dem Beweis von Lemma 7.27 dabei ein Arbeitsfaktor von $\alpha(2,3-\varepsilon)$, sodass in diesem Fall insgesamt ein Arbeitsfaktor von $\alpha(3,4-\varepsilon,4)$ vorliegt.

Im zweiten Fall bilden $(v,1)$ und $(w,1)$ ein Dreieck mit $(x,1)$, und $(x,1)$ kommt in drei Einschränkungen vor (siehe Abb. 7.25(b)). Wie im Beweis des zweiten Falls von Lemma 7.33 können wir eines der drei Paare $(v,1)$, $(w,1)$ und $(x,1)$ wählen. Weil auch alle Nachbarvariablen von $(x,1)$ nur drei Farben haben (denn sonst wäre Lemma 7.33 auf I anwendbar gewesen), ergibt sich hier ein Arbeitsfaktor von $\alpha(3,4,4)$.

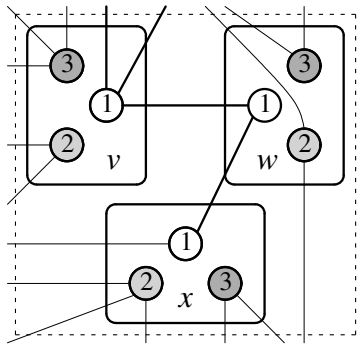
Im dritten Fall schließlich bilden $(v,1)$ und $(w,1)$ ein Dreieck mit $(x,1)$, und $(x,1)$ kommt diesmal aber nur in zwei Einschränkungen vor (siehe Abb. 7.25(c)). Wählen wir die Farbe 1 für v, so eliminieren wir wieder vier Variablen; schließen wir die Wahl von $(v,1)$ jedoch aus, so entsteht eine isolierte Einschränkung zwischen $(w,1)$ und $(x,1)$. In diesem Fall ist der Arbeitsfaktor $\alpha(1+\varepsilon,4)$.

Für alle drei Fälle ergibt sich ein Arbeitsfaktor von höchstens

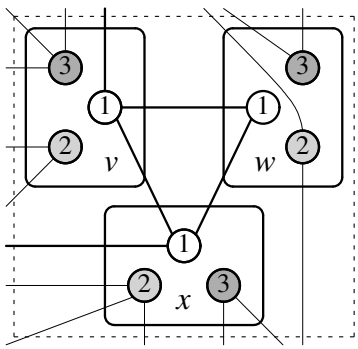
$$\max\{\alpha(1+\varepsilon,4),\alpha(3,4-\varepsilon,4)\},$$

wie im Lemma behauptet. ❑

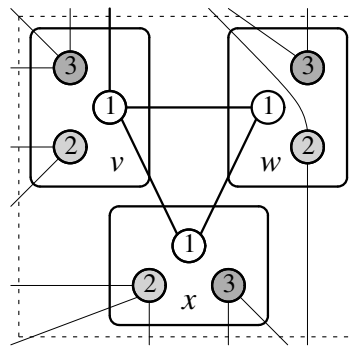
Was haben wir bisher geschafft? Um eine Zwischenbilanz des Erreichten zu geben: Wenn keines der oben angegebenen Lemmata 7.13, 7.25, 7.27, 7.30, 7.31, 7.33



(a) $(v,1)$, $(w,1)$ und $(x,1)$ bilden kein Dreieck



(b) $(v,1)$, $(w,1)$ und $(x,1)$ bilden ein Dreieck, $(x,1)$ kommt in drei Einschränkungen vor



(c) $(v,1)$, $(w,1)$ und $(x,1)$ bilden ein Dreieck, $(x,1)$ kommt in zwei Einschränkungen vor

Abb. 7.25. Drei Fälle im Beweis von Lemma 7.35

und 7.35 auf eine gegebene $(4,2)$-CSP-Instanz mehr anwendbar ist, dann muss jedes Paar (v,i) (wobei v eine Variable und i eine Farbe ist) an entweder zwei oder drei Einschränkungen beteiligt sein, und jedes Nachbarpaar von (v,i) muss derselben Anzahl von Einschränkungen wie (v,i) unterworfen sein.

Dreifach eingeschränkte Farben

Noch sind wir nicht mit der Beschreibung der vielen Möglichkeiten fertig, die zur Vereinfachung von $(4,2)$-CSP-Instanzen dienen können, doch das Ende ist schon in Sicht: Wenn wir uns in den folgenden Lemmata um die oben erwähnten dreifach und zweifach eingeschränkten Farben gekümmert haben, werden alle relevanten lokalen Situationen abgearbeitet sein. Uns werden dann so einfache Instanzen vorliegen, dass wir sie mit einem einfachen Matching-Algorithmus schnell lösen können.

Wir nehmen im Folgenden an, dass keines der bisherigen Lemmata auf unsere gegebene $(4,2)$-CSP-Instanz I mehr anwendbar ist; also ist jedes Paar (v,i) an genau so vielen Einschränkungen beteiligt wie jedes seiner Nachbarpaare. Betrachten wir

Dreierkomponente

zunächst die dreifach eingeschränkten Farben. Eine *Dreierkomponente* ist eine Menge D von Paaren (v,i), sodass jedes Paar in D zu drei Einschränkungen gehört und mit jedem anderen Paar in D durch eine Folge von Einschränkungen verbunden ist.[43] Es ist zweckmäßig, große von kleinen Dreierkomponenten zu unterscheiden. Wir sagen,

kleine bzw. große Dreierkomponente

eine Dreierkomponente D ist *klein*, falls nur vier Variablen an D beteiligt sind, und D ist *groß*, falls fünf oder mehr Variablen an D beteiligt sind. Dabei nehmen wir an, dass sämtlichen Variablen in jeder Dreierkomponente nur drei Farben zur Verfügung stehen (andernfalls könnten wir eines der oben angegebenen Lemmata anwenden).

Lemma 7.36 (Beigel und Eppstein [BE05]). *Ist D eine kleine Dreierkomponente in einer $(4,2)$-CSP-Instanz, so ist $k = |D|$ ein Vielfaches von vier, und jede in D vorkommende Variable taucht in genau $k/4$ der Paare in D auf.* **ohne Beweis**

Übung 7.37. Beweisen Sie Lemma 7.36.

gute Dreierkomponente

Hat eine kleine Dreierkomponente genau vier Elemente, so bezeichnen wir sie als *gut*, denn dann braucht sie nicht weiter vereinfacht zu werden. Das nächste Lemma vereinfacht solche kleinen Dreierkomponenten, die noch nicht gut sind.

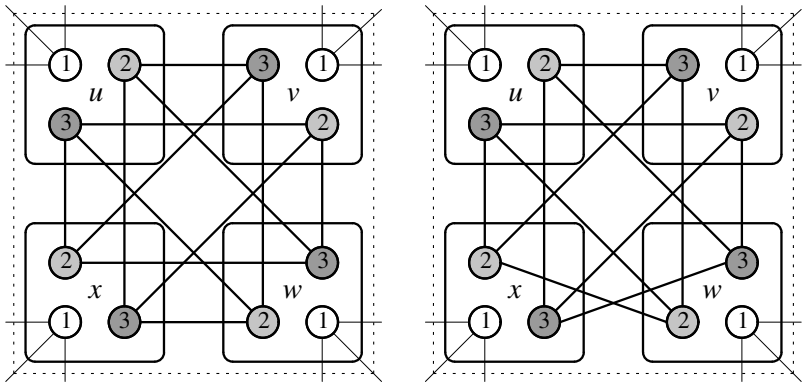
Lemma 7.38 (Beigel und Eppstein [BE05]). *Jede $(4,2)$-CSP-Instanz I mit einer kleinen Dreierkomponente D, die nicht gut ist, kann durch kleinere Instanzen mit Arbeitsfaktor $\alpha(4,4,4)$ ersetzt werden, sodass I genau dann lösbar ist, wenn mindestens eine dieser kleineren Instanzen lösbar ist.*

Beweis. Eine kleine Dreierkomponente D der Größe $k = 12$ braucht alle Farbwahlmöglichkeiten für sämtliche vier Variablen auf. Deshalb können wir sie unabhängig von der restlichen Instanz betrachten: Wenn möglich, färben wir diese Variablen so, dass alle beteiligten Einschränkungen erfüllt werden, und entfernen diese vier 3-Farben-Variablen; andernfalls ist die Instanz I nicht lösbar.

Andernfalls muss D genau $k = 8$ Paare enthalten, da D nicht gut ist. Man kann sich leicht überlegen, dass (bis auf Umbenennen der Farben) nur die in Abb. 7.26 dargestellten zwei Möglichkeiten für D in Frage kommen (siehe Übung 7.39 unten).

Im Fall von Abb. 7.26(a) färben wir alle vier Variablen so (entweder alle mit 2 oder alle mit 3), dass die beteiligten Einschränkungen erfüllt sind und diese vier Variablen von der Instanz entfernt werden können. Im Fall von Abb. 7.26(b) dagegen gibt es drei maximale Teilmengen von Variablen, die gefärbt werden können: entweder $\{u,v,w\}$ oder $\{u,v,x\}$ oder $\{w,x\}$. Entsprechend unterteilen wir in kleinere Instanzen, und da das Färben von Variablen in den jeweiligen Teilmengen Farben der übrigen Variablen eliminiert, können wir auch hier mit der letzten Aussage von Lemma 7.13 alle vier Variablen entfernen. Da wir in jedem Fall vier 3-Farben-Variablen eliminieren, ergibt sich ein Arbeitsfaktor von $\alpha(4,4,4)$. ❑

[43] Dies entspricht dem Begriff der Zusammenhangskomponente in einem Graphen, welche nach Definition 3.6 dadurch gekennzeichnet ist, dass je zwei Knoten durch einen Pfad miteinander verbunden sind.



(a) Erste Möglichkeit (b) Zweite Möglichkeit

Abb. 7.26. Die zwei möglichen kleinen Dreierkomponenten mit acht Paaren

Übung 7.39. Zeigen Sie, dass – wie im Beweis von Lemma 7.38 behauptet – für eine kleine Dreierkomponente mit acht Paaren die beiden in Abb. 7.26 dargestellten $(3,2)$-CSP-Instanzen die einzig möglichen sind.

Nun wenden wir uns den großen Dreierkomponenten zu. Die erste wichtige Beobachtung ist, dass wir in jeder großen Dreierkomponente eine bestimmte Struktur finden können, die wir als einen *Zeugen* dieser Dreierkomponente bezeichnen. Ein solcher Zeuge besteht aus fünf Paaren (v,i), (w,i), (x,i), (y,i) und (z,i), sodass gilt:

Zeuge einer großen Dreierkomponente

1. u, v, w, x, y und z sind verschiedene Variablen,
2. es gibt die Einschränkungen $((v,i),(w,i))$, $((v,i),(x,i))$ und $((v,i),(y,i))$ und
3. mindestens eines der Paare (w,i), (x,i) und (y,i) bildet eine Einschränkung mit (z,i).

Lemma 7.40 (Beigel und Eppstein [BE05]). *Jede große Dreierkomponente in einer* $(4,2)$-CSP-*Instanz hat einen Zeugen.* **ohne Beweis**

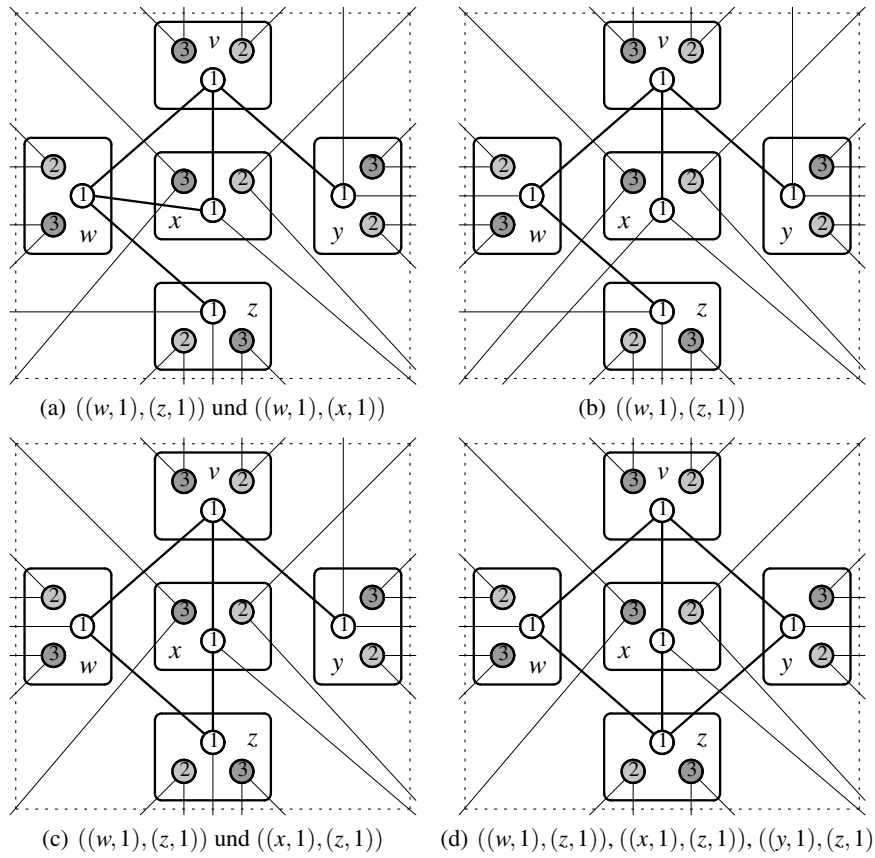
Übung 7.41. Zeigen Sie, dass jede große Dreierkomponente in einer $(4,2)$-CSP-Instanz einen Zeugen hat, d. h., beweisen Sie Lemma 7.40.
Hinweis: Fassen Sie die $(4,2)$-CSP-Instanz als einen Graphen auf, dessen Knoten die Farben in den einzelnen Variablen sind (also Paare der Form (v,j)) und dessen Kanten die Einschränkungen sind. Ausgehend von einem beliebig gewählten Paar (u,i) kann nun eine Breitensuche (siehe Abschnitt 3.3.2 und Abb. 3.18) auf diesem Graphen ausgeführt werden, um einen Zeugen zu finden.

Das folgende Lemma zeigt, wie man $(4,2)$-CSP-Instanzen mit großen Dreierkomponenten vereinfachen kann.

Lemma 7.42 (Beigel und Eppstein [BE05]). *Jede* $(4,2)$-CSP-*Instanz I mit einer großen Dreierkomponente kann durch kleinere Instanzen mit einem Arbeitsfaktor von* $\alpha(4,4,5,5)$ *ersetzt werden, sodass I genau dann lösbar ist, wenn mindestens eine dieser kleineren Instanzen lösbar ist.*

Beweis. Sei I eine $(4,2)$-CSP-Instanz mit einer großen Dreierkomponente D. Nach Lemma 7.40 gibt es einen Zeugen von D, dieser sei durch die fünf Paare $(v,1)$, $(w,1)$, $(x,1)$, $(y,1)$ und $(z,1)$ gegeben. Abhängig davon, wie viele der Paare $(w,1)$, $(x,1)$ und $(y,1)$ eine Einschränkung mit $(z,1)$ bilden, betrachten wir die folgenden Fälle:

1. Nur eines der Paare $(w,1)$, $(x,1)$ und $(y,1)$ bildet mit $(z,1)$ eine Einschränkung, etwa $((w,1),(z,1))$ (Abb. 7.27(a) und Abb. 7.27(b)).
2. Zwei der Paare $(w,1)$, $(x,1)$ und $(y,1)$ bilden mit $(z,1)$ eine Einschränkung, zum Beispiel $((w,1),(z,1))$ und $((x,1),(z,1))$ (Abb. 7.27(c)).
3. Alle drei Paare $(w,1)$, $(x,1)$ und $(y,1)$ bilden mit $(z,1)$ eine Einschränkung, nämlich $((w,1),(z,1))$, $((x,1),(z,1))$ und $((y,1),(z,1))$ (Abb. 7.27(d)).



(a) $((w,1),(z,1))$ und $((w,1),(x,1))$ (b) $((w,1),(z,1))$

(c) $((w,1),(z,1))$ und $((x,1),(z,1))$ (d) $((w,1),(z,1))$, $((x,1),(z,1))$, $((y,1),(z,1))$

Abb. 7.27. Fallunterscheidung für den Zeugen einer großen Dreierkomponente in Lemma 7.42

Im ersten Fall, wenn etwa $((w,1),(z,1))$ die eine Einschränkung zwischen $(z,1)$ und einem der Paare $(w,1)$, $(x,1)$ und $(y,1)$ ist, können wir entweder z mit 1 färben

oder uns dagegen entscheiden. Färben wir z mit 1, so wird z unmittelbar eliminiert. Außerdem ist dann eine von drei Farben in den drei Variablen (einschließlich w) ausgeschlossen, die mit $(z,1)$ durch eine Einschränkung verbunden sind, sodass auch diese drei Variablen mit der letzten Aussage von Lemma 7.13 entfernt werden können; insgesamt haben wir die Größe der Instanz dann also um vier 3-Farben-Variablen verringert.

Entscheiden wir uns im ersten Fall jedoch dagegen, z mit 1 zu färben, so ist das Paar $(w,1)$ nur noch an zwei Einschränkungen beteiligt. Die eine ist $((v,1),(w,1))$. Ist an der anderen Einschränkung mit $(w,1)$ eines der Paare $(x,1)$ oder $(y,1)$ beteiligt, so bildet dieses Paar mit $(v,1)$ und $(w,1)$ ein Dreieck (siehe Abb. 7.27(a)). Wir können ohne Beschränkung der Allgemeinheit davon ausgehen, dass die „noch freie“ Einschränkung an $(x,1)$ bzw. $(y,1)$ (in Abb. 7.27(a) ist dies die Einschränkung an $(x,1)$, die nicht mit $(v,1)$ oder $(w,1)$ verbunden ist) nicht mit einer anderen Farbe von z verbunden ist, denn andernfalls könnten wir die Variablen v, w, x, y und z so umsortieren, dass ein Zeuge der gewünschten Form entsteht.[44]

Dieses Dreieck mit den Ecken $(v,1)$, $(w,1)$ und $(x,1)$ kann dann so wie das Dreieck aus Abb. 7.25(b) behandelt werden: Wir unterteilen in kleinere Instanzen, je nachdem, ob wir die Färbung $(v,1)$, $(w,1)$ oder $(x,1)$ auswählen. Der Arbeitsfaktor, der sich gemäß dem zweiten Fall im Beweis von Lemma 7.35 ergibt (siehe Abb. 7.25(b)), wäre $\alpha(3,4,4)$. Da zusätzlich die 3-Farben-Variable z eliminiert wird, erhalten wir hier stattdessen sogar $\alpha(4,5,5)$. Zusammen mit der oben genannten Entfernung von vier 3-Farben-Variablen, falls z mit 1 gefärbt wird, ergibt sich insgesamt ein Arbeitsfaktor $\alpha(4,4,5,5)$.

Der noch fehlende Unterfall des ersten Falls (nämlich dass wir z nicht mit 1 färben, aber $(w,1)$ weder mit $(x,1)$ noch mit $(y,1)$ eine Einschränkung bildet; siehe Abb. 7.27(b)) liefert denselben Arbeitsfaktor $\alpha(4,4,5,5)$. Übung 7.43 unten überlässt dem Leser die detaillierte Diskussion dieses Unterfalls.

Im zweiten Fall bilden zwei der Paare $(w,1)$, $(x,1)$ und $(y,1)$ mit $(z,1)$ eine Einschränkung, sagen wir, $((w,1),(z,1))$ und $((x,1),(z,1))$ (siehe Abb. 7.27(c)). Färben wir nun z mit 1, so eliminieren wir vier 3-Farben-Variablen (nämlich z, w, x und den dritten Nachbarn von $(z,1)$) und lassen $(v,1)$ folglich baumeln. Da wir nur noch 3-Farben-Variablen betrachten, ändert sich der Arbeitsfaktor $\alpha(2,3-\varepsilon)$ (siehe (7.9) im Beweis von Lemma 7.27) zu $\alpha(2,3)$, und zusammen mit den oben erwähnten vier bereits eliminierten Variablen ergibt sich $\alpha(6,7)$. Entscheiden wir uns jedoch im zweiten Fall, z nicht mit 1 zu färben, so verringern wir die Größe der Instanz lediglich um eine Variable, nämlich z. Insgesamt ergibt sich im zweiten Fall somit ein Arbeitsfaktor von $\alpha(1,6,7)$.

Im dritten Fall schließlich ist $(z,1)$ mit jedem der Paare $(w,1)$, $(x,1)$ und $(y,1)$ durch eine Einschränkung verbunden (siehe Abb. 7.27(d)). Färben wir z mit 1, so können wir nicht nur z, w, x und y eliminieren, sondern sind auch in der Lage, die Farbe 1 für v zu verwenden. Somit sind wir fünf Variablen losgeworden. Andererseits verringert das Vermeiden der Farbe 1 für z die Größe der Instanz nur um eine

[44] Genauer gesagt spielt w dann die Rolle von v; y die von z; und v, x und z spielen die Rolle von w, x und y.

Variable, nämlich wieder z. Es ergibt sich im dritten Fall also ein Arbeitsfaktor von $\alpha(1,5)$.

Unter den Arbeitsfaktoren in diesen drei Fällen stellt sich der erste, $\alpha(4,4,5,5)$, als der größte heraus. ❑

Übung 7.43. Zeigen Sie, dass sich für den der Abb. 7.27(b) im Beweis von Lemma 7.42 entsprechenden Fall ebenfalls ein Arbeitsfaktor von $\alpha(4,4,5,5)$ ergibt.
Hinweis: Schließen Sie die Färbung $(z,1)$ aus und gehen Sie dann wie im ersten Fall des Beweises von Lemma 7.35 vor (siehe Abb. 7.25(a)).

Doppelt eingeschränkte Farben

Nun haben wir es fast geschafft! Die letzte Möglichkeit zur Vereinfachung von $(4,2)$-CSP-Instanzen, die hier beschrieben werden soll, betrifft solche Paare (v,i), die an genau zwei Einschränkungen beteiligt sind. Analog zu den Dreierkomponenten bezeichnen wir eine Menge solcher Paare, in der jedes Paar mit jedem anderen Paar über eine Folge von Einschränkungen verbunden ist, als eine *Zweierkomponente* der Instanz. Wieder unterscheiden wir zwischen kleinen und großen Zweierkomponenten. Zu einer *kleinen Zweierkomponente* gehören nur drei Paare; zu einer *großen Zweierkomponente* gehören vier oder mehr Paare.

Zweierkomponente

kleine bzw. große Zweierkomponente

Zunächst stellen wir fest, dass eine Zweierkomponente stets einen Zyklus von Paaren bilden muss. Dabei ist es möglich, dass dieselbe Variable in mehr als einem Paar des Zyklus vorkommt. Anders als bei den dreifach eingeschränkten Farben können wir hier nicht annehmen, dass allen Variablen nur drei Farben zur Auswahl stehen, denn da die Paare einer Zweierkomponente stets in zwei Einschränkungen auftreten, sind manche der früheren Lemmata (wie Lemma 7.31, Lemma 7.33 und Lemma 7.35) nicht anwendbar.

Lemma 7.44 (Beigel und Eppstein [BE05]). *Sei $\varepsilon \leq 0.287$. Jede $(4,2)$-CSP-Instanz I mit einer großen Zweierkomponente kann durch kleinere Instanzen mit Arbeitsfaktor $\alpha(3,3,5)$ ersetzt werden, sodass I genau dann lösbar ist, wenn mindestens eine dieser kleineren Instanzen lösbar ist.*

Beweis. Für eine gegebene große Zweierkomponente D der $(4,2)$-CSP-Instanz I unterscheiden wir die folgenden drei Fälle:

1. Der D entsprechende Zyklus durchläuft fünf aufeinander folgende Paare mit verschiedenen Variablen, etwa die Paare $(v,1)$, $(w,1)$, $(x,1)$, $(y,1)$ und $(z,1)$, mit $|\{v,w,x,y,z\}| = 5$.
2. Der D entsprechende Zyklus enthält im Abstand von drei Einschränkungen zwei Paare mit derselben Variablen; beispielsweise könnte D die Paare $(v,1)$, $(w,1)$, $(x,1)$ und $(v,2)$ enthalten, und es gibt die Einschränkungen $((v,1),(w,1))$, $((w,1),(x,1))$ und $((x,1),(v,2))$.
3. Keiner der ersten beiden Fälle tritt ein.

Im ersten Fall können wir ausschließen, dass z die einzige 4-Farben-Variable unter den fünf Variablen v, w, x, y und z ist, denn andernfalls könnten wir diese fünf Variablen umbenennen und in umgekehrter Reihenfolge auflisten. Das heißt also, wenn überhaupt eine dieser fünf Variablen vier Farben zur Auswahl hat, dann ist dies auch für eine der Variablen v, w, x und y so.

Nehmen wir erst einmal an, dass allen fünf Variablen nur drei Farben zur Verfügung stehen. Jede Färbung, die alle Einschränkungen der Instanz I erfüllt, aber nicht sowohl v als auch y mit 1 färbt, kann ohne Verletzung von Einschränkungen in eine solche Lösung von I abgewandelt werden, die entweder w oder x mit 1 färbt. Somit können wir I zu einer der folgenden drei kleineren Instanzen vereinfachen:

- Färben wir w mit 1, so wird w entfernt und v und x (die beiden Nachbarn von w) werden um je eine Farbe ärmer, wodurch sie ebenfalls eliminiert werden können. Die resultierende Instanz hat also drei 3-Farben-Variablen weniger.
- Färben wir x mit 1, so können wir wieder drei 3-Farben-Variablen entfernen, nämlich w, x und y.
- Färben wir sowohl v als auch y mit 1, so werden wir auf einen Schlag alle fünf 3-Farben-Variablen los.

Der Arbeitsfaktor in diesem Fall ist also $\alpha(3,3,5)$. Verfügt mindestens eine der fünf Variablen v, w, x, y und z über vier Farben, so erhalten wir mit der entsprechenden Aufteilung einen Arbeitsfaktor von höchstens $\alpha(3-\varepsilon,4-\varepsilon,5-2\varepsilon)$, der für $\varepsilon \leq 0.287$ kleiner als $\alpha(3,3,5)$ und somit irrelevant ist.

Im zweiten Fall nehmen wir an, dass dieselbe Variable in zwei Paaren unseres Zyklus vorkommt, die drei Einschränkungen weit auseinander liegen; sagen wir, zum Zyklus gehören die Paare $(v,1)$, $(w,1)$, $(x,1)$ und $(v,2)$ mit den Einschränkungen $((v,1),(w,1))$, $((w,1),(x,1))$ und $((x,1),(v,2))$. Jede Färbung, die alle Einschränkungen von I erfüllt, kann ohne Verletzung von Einschränkungen so abgeändert werden, dass entweder w oder x mit 1 gefärbt wird. Färben wir einerseits w mit 1 bzw. andererseits x mit 1, so erhalten wir in beiden Fällen eine um höchstens $3-\varepsilon$ kleinere Instanz. Somit ist der Arbeitsfaktor im zweiten Fall höchstens $\alpha(3-\varepsilon,3-\varepsilon)$. Der schlechteste Fall tritt dabei ein, wenn nur v vier Farben zur Verfügung stehen und den anderen vier Variablen jeweils drei.

Zuletzt nehmen wir an, dass weder der erste noch der zweite Fall für unseren Zyklus eintritt. Dann muss dieser einmal, zweimal oder dreimal dieselben vier Variablen in derselben Reihenfolge durchlaufen. Wir überlassen es dem Leser, sich zu überlegen, weshalb sich in diesem Fall der Arbeitsfaktor $\alpha(4,4)$ ergibt (Übung 7.45 unten).

Für $\varepsilon \leq 0.287$ ist $\alpha(3,3,5)$ der dominante Arbeitsfaktor dieser drei Fälle. ❑

Übung 7.45. Zeigen Sie, dass sich im dritten Fall des Beweises von Lemma 7.44 ein Arbeitsfaktor von $\alpha(4,4)$ ergibt.

Matching

Lässt sich keines der oben in den Abschnitten 7.4.1 und 7.4.3 angegebenen Lemmata (also weder Lemma 7.13 noch eines der Lemmata 7.25 bis 7.35, 7.38, 7.42 und 7.44) auf eine $(4,2)$-CSP-Instanz mehr anwenden, so muss diese eine ganz einfache Struktur haben: Jede Einschränkung muss dann nämlich entweder zu einer guten Dreierkomponente oder zu einer kleinen Zweierkomponente gehören. Zur Erinnerung: In einer guten Dreierkomponente D gibt es vier Paare der Form (v,i), wobei v eine Variable und i eine Farbe ist, sodass jedes Paar in D mit jedem anderen Paar in D durch eine Einschränkung verbunden ist. Eine kleine Zweierkomponente Z besteht aus drei Paaren, sodass jedes Paar in Z mit jedem anderen Paar in Z durch eine Einschränkung verbunden ist. Abbildung 7.28 zeigt eine $(3,2)$-CSP-Instanz mit den folgenden drei guten Dreierkomponenten:

$$\begin{aligned} D_1 &= \{(u_1,1),(v_1,1),(w_1,1),(x_1,1)\},\\ D_2 &= \{(u_2,1),(v_2,1),(w_2,1),(x_2,1)\},\\ D_3 &= \{(u_3,1),(v_3,1),(w_3,1),(x_3,1)\} \end{aligned} \tag{7.13}$$

und den folgenden acht kleinen Zweierkomponenten:

$$\begin{aligned} Z_1 &= \{(u_1,3),(v_1,3),(w_1,3)\}, & Z_2 &= \{(v_1,2),(w_1,2),(x_1,2)\},\\ Z_3 &= \{(u_2,3),(v_2,3),(w_2,3)\}, & Z_4 &= \{(v_2,2),(w_2,2),(x_2,2)\},\\ Z_5 &= \{(u_3,3),(v_3,3),(w_3,3)\}, & Z_6 &= \{(v_3,2),(w_3,2),(x_3,2)\},\\ Z_7 &= \{(u_1,2),(u_2,2),(x_1,3)\}, & Z_8 &= \{(u_3,2),(x_2,3),(x_3,3)\}. \end{aligned} \tag{7.14}$$

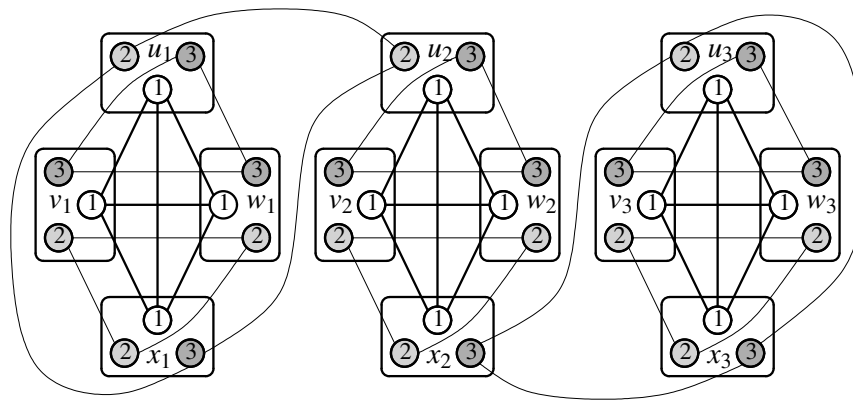


Abb. 7.28. Instanz mit drei guten Dreierkomponenten und acht kleinen Zweierkomponenten

Eine so einfach strukturierte Instanz kann nun schnell durch einen Matching-Algorithmus (siehe Abschnitt 3.3.5) gelöst werden.

Lemma 7.46 (Beigel und Eppstein [BE05]). *Für eine gegebene* $(4,2)$-CSP-*Instanz, in der jede Einschränkung entweder zu einer guten Dreierkomponente oder einer*

kleinen Zweierkomponente gehört, kann man in Polynomialzeit entscheiden, ob sie lösbar ist (und, falls ja, eine Lösung angeben).

Beweis. Wir führen den Beweis anhand der Beispielinstanz I in Abb. 7.28. Zunächst definieren wir ausgehend von I einen bipartiten Graphen $G = (V,E)$ wie folgt:

- Die Knotenmenge besteht aus den Variablen von I einerseits und aus den guten Dreierkomponenten und kleinen Zweierkomponenten von I andererseits. Im Beispiel von Abb. 7.28 erhalten wir also die Knotenmenge $V = V_1 \cup V_2$, die definiert ist durch

$$V_1 = \{u_i, v_i, w_i, x_i \mid 1 \leq i \leq 3\} \text{ und}$$
$$V_2 = \{D_1, D_2, D_3\} \cup \{Z_1, Z_2, \ldots, Z_8\},$$

  mit den in (7.13) und (7.14) angegebenen Dreierkomponenten D_i und Zweierkomponenten Z_j.
- Eine Kante existiert genau dann zwischen einem Variablenknoten $y \in V_1$ und einem Komponentenknoten $z \in V_2$, wenn ein Paar (y,i) in der Komponente z vorkommt. Im Beispiel von Abb. 7.28 erhalten wir also die Kantenmenge

$$\begin{aligned} E = {} & \{\{u_i, D_i\}, \{v_i, D_i\}, \{w_i, D_i\}, \{x_i, D_i\} \mid 1 \leq i \leq 3\} \cup \\ & \{\{u_1, Z_1\}, \{v_1, Z_1\}, \{w_1, Z_1\}\} \cup \{\{v_1, Z_2\}, \{w_1, Z_2\}, \{x_1, Z_2\}\} \cup \\ & \{\{u_2, Z_3\}, \{v_2, Z_3\}, \{w_2, Z_3\}\} \cup \{\{v_2, Z_4\}, \{w_2, Z_4\}, \{x_2, Z_4\}\} \cup \\ & \{\{u_3, Z_5\}, \{v_3, Z_5\}, \{w_3, Z_5\}\} \cup \{\{v_3, Z_6\}, \{w_3, Z_6\}, \{x_3, Z_6\}\} \cup \\ & \{\{u_1, Z_7\}, \{u_2, Z_7\}, \{x_1, Z_7\}\} \cup \{\{u_3, Z_8\}, \{x_2, Z_8\}, \{x_3, Z_8\}\}. \end{aligned}$$

Abbildung 7.29 zeigt den bipartiten Graphen, der gemäß der oben angegebenen Konstruktion aus der in Abb. 7.28 dargestellten $(3,2)$-CSP-Instanz I entstanden ist.

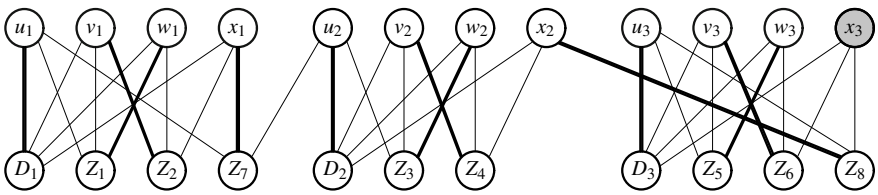


Abb. 7.29. Ein nicht perfektes Matching in einem bipartiten Graphen für Lemma 7.46

Sowohl gute Dreierkomponenten als auch kleine Zweierkomponenten haben die Eigenschaft, dass jedes in ihnen enthaltene Paar mit jedem anderen Paar derselben Komponente durch eine Einschränkung verbunden ist. Eine Lösung der Instanz ist also eine Menge solcher Paare, in der jede Variable genau einmal vorkommt und jede Komponente höchstens ein Paar enthält. Anders gesagt gibt es genau dann eine

Lösung der Instanz, wenn der oben konstruierte bipartite Graph ein perfektes Matching besitzt, eine Teilmenge M von E also, sodass je zwei Kanten in M disjunkt sind (d. h., keine zwei Kanten sind mit demselben Knoten inzident) und sodass alle Knoten des Graphen (insbesondere alle Variablenknoten) Endpunkt einer Kante in M sind.

Um zu testen, ob die gegebene $(4,2)$-CSP-Instanz I lösbar ist, wenden wir also einfach einen Polynomialzeit-Algorithmus an, der bestimmt, ob der bipartite Graph ein perfektes Matching besitzt. In Abb. 7.29 ist ein maximales Matching durch fettgedruckte Kanten angegeben, das der folgenden partiellen Färbung von I entspricht:

$$(u_1,1),\ (v_1,2),\ (w_1,3),\ (x_1,3),\ (u_2,1),\ (v_2,2),\\ (w_2,3),\ (x_2,3),\ (u_3,1),\ (v_3,2),\ (w_3,3).$$

Beispielsweise bewirkt die Färbung $(u_1,1)$ eine Kante zwischen u_1 und D_1 im Matching, weil $(u_1,1)$ zu D_1 gehört. Die einzige noch ungefärbte Variable, x_3, kann nun aber nicht mehr gefärbt werden, ohne eine der Einschränkungen

$$((u_3,1),(x_3,1)),\quad ((v_3,2),(x_3,2))\quad \text{oder}\quad ((x_2,3),(x_3,3))$$

zu verletzen. In Abb. 7.29 entspricht dies der Tatsache, dass sämtliche mit dem schattierten Variablenknoten x_3 verbundenen Komponentenknoten bereits mit einer Kante im Matching inzident sind. Dass es in diesem bipartiten Graphen kein perfektes Matching geben kann, sieht man schon daran, dass es mehr Variablenknoten als Komponentenknoten gibt. Demgemäß ist die in Abb. 7.28 dargestellte $(4,2)$-CSP-Instanz I nicht lösbar. ❑

Laufzeit des deterministischen CSP-Algorithmus

Abbildung 7.30 stellt den deterministischen Algorithmus für das Problem $(3,2)$-CSP bzw. $(4,2)$-CSP im Überblick dar, dessen Laufzeit wir nun analysieren wollen.

Wie man sieht, kommen im ersten und dritten Schritt dieses Algorithmus keine rekursiven Aufrufe vor, wohl aber im zweiten. Abbildung 7.31 stellt einen konkreten Rekursionsbaum für diesen zweiten Schritt des Algorithmus dar. Die Wurzel des Baumes ist eine bereits gemäß Lemma 7.13 reduzierte Instanz. Seine inneren Knoten geben das jeweilige Lemma an, das sich auf die aktuelle Instanz anwenden lässt, wobei die Kinder eines inneren Knotens die gemäß diesem Lemma vereinfachten Instanzen enthalten und die Kanten zu diesen Kindern mit der jeweils erzielten Reduzierung der Instanzengröße beschriftet sind. Die Blätter des Rekursionsbaumes schließlich enthalten solche Instanzen, auf die keines der Lemmata 7.25 bis 7.35, 7.38, 7.42 und 7.44 mehr anwendbar ist. Die inneren Knoten des Rekursionsbaumes repräsentieren also die rekursiven Aufrufe des Algorithmus, während die Blätter solche Instanzen darstellen, die ohne weitere rekursive Aufrufe mit dem Matching-Algorithmus aus Lemma 7.46 gelöst werden können. Findet dieser ein perfektes Matching in dem Graphen, der der Instanz in einem der Blätter entspricht, so taucht man Schritt für Schritt aus den Rekursionen wieder auf, erhält schließlich eine

1. Solange die gegebene Instanz noch nicht reduziert ist, wende Lemma 7.13 an.
2. Solange eines der Lemmata 7.25 bis 7.35, 7.38, 7.42 und 7.44 auf die aktuelle Instanz anwendbar ist, unterteile die Instanz gemäß dem jeweiligen Lemma in die entsprechende Anzahl kleinerer Instanzen und löse diese rekursiv. Findet irgendein rekursiver Aufruf eine Lösung seiner Instanz, so übersetze diese zurück in eine Lösung der übergeordneten Instanz (von der der rekursive Aufruf ausging) und gib diese Lösung zurück. Andernfalls gib eine Meldung zurück, dass diese Instanz nicht lösbar ist.
3. Ist keines der Lemmata 7.25 bis 7.35, 7.38, 7.42 und 7.44 mehr auf die aktuelle Instanz anwendbar, so muss jede ihrer Einschränkungen entweder zu einer guten Dreierkomponente oder zu einer kleinen Zweierkomponente gehören. Konstruiere aus dieser Instanz einen bipartiten Graphen gemäß Lemma 7.46 und wende den dort erwähnten Polynomialzeit-Algorithmus zur Bestimmung eines maximalen Matching auf diesen Graphen an. Ist dieses Matching perfekt, übersetze es zurück in eine Färbung der Instanz. Andernfalls gib eine Meldung zurück, dass diese Instanz nicht lösbar ist.

Abb. 7.30. Deterministischer Algorithmus für $(3,2)$-CSP bzw. $(4,2)$-CSP.

Lösung der ursprünglich gegebenen Instanz und beendet die Suche. Andernfalls setzt man die Suche im Sinne einer Backtracking-Strategie an der letzten Verzweigung im Rekursionsbaum fort.

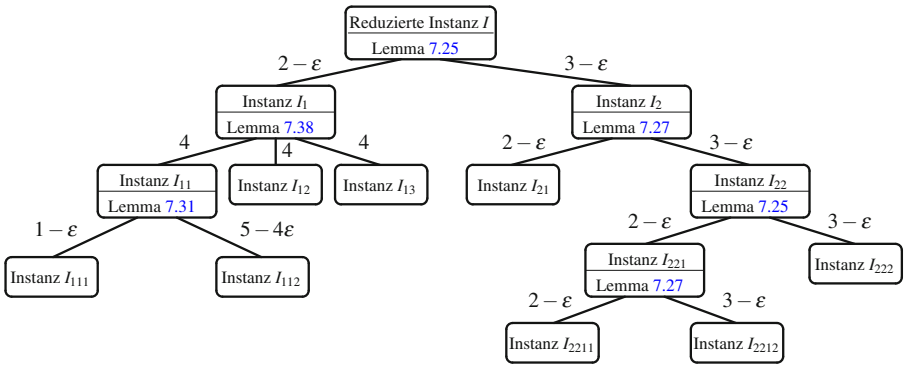


Abb. 7.31. Rekursionsbaum des deterministischen CSP-Algorithmus aus Abb. 7.30

Die Korrektheit des Algorithmus in Abb. 7.30 folgt unmittelbar aus den oben angegebenen Lemmata, denn diese zeigen, wie man eine gegebene Instanz I in kleinere Instanzen I_j überführt, von denen mindestens eine zu I äquivalent ist. Wenn I also lösbar ist, so findet man in wenigstens einem Blatt des Rekursionsbaumes eine Lösung der entsprechenden kleineren Instanz, und der Algorithmus akzeptiert seine Eingabe (und liefert als Lösung eine Färbung der Variablen, die alle Einschränkungen erfüllt). Ist I jedoch nicht lösbar, so kann keines der Blätter im Rekursionsbaum

eine lösbare kleinere Instanz enthalten, und der Algorithmus muss seine Eingabe verwerfen, nachdem der gesamte Rekursionsbaum erfolglos durchsucht wurde.

Der folgende Satz analysiert die Laufzeit dieses Algorithmus.

Satz 7.47 (Beigel und Eppstein [BE05]). *Der Algorithmus in Abb. 7.30 löst*

- *das Problem* $(3,2)$-CSP *in der Zeit* $\tilde{\mathcal{O}}(1.36443^n)$ *und*
- *das Problem* $(4,2)$-CSP *in der Zeit* $\tilde{\mathcal{O}}(1.80721^n)$.

Beweis. Die Größe einer gegebenen $(4,2)$-CSP-Instanz I ist $n = n_3 + (2-\varepsilon)n_4$ (siehe (7.7)), wobei n_3 bzw. n_4 die Anzahl der Variablen mit 3 bzw. 4 Farben in I bezeichnet. Ist I dagegen eine $(3,2)$-CSP-Instanz, so bezeichnet n einfach die Anzahl der Variablen von I, sodass sich eine als Funktion von n dargestellte Schranke für die Laufzeit des Algorithmus in Abb. 7.30 unmittelbar auf $(3,2)$-CSP bezieht.

Wir beginnen mit der Analyse der Laufzeit des Algorithmus in Abb. 7.30 für das Problem $(3,2)$-CSP. Wie zu Beginn von Abschnitt 7.4.3 erklärt wurde, ist $\tilde{\mathcal{O}}(\alpha^n)$ eine obere Schranke für diese Laufzeit, wobei α der größte Arbeitsfaktor bezüglich der lokalen Situationen ist, die wir in den obigen Lemmata identifiziert haben. Das heißt, die Anzahl der rekursiven Aufrufe wird durch α^n dominiert, und im Vergleich damit sind die polynomialzeit-beschränkten Phasen des Algorithmus vernachlässigbar, die sich etwa bei der Suche nach verwertbaren lokalen Situationen und bei der Ausführung der entsprechenden Transformationen in kleinere Instanzen sowie bei der Anwendung von Lemma 7.13 im ersten Schritt des Algorithmus und von Lemma 7.46 im dritten Schritt ergeben.

Tabelle 7.3 fasst die einzelnen Arbeitsfaktoren aus den Lemmata 7.25 bis 7.35, 7.38, 7.42 und 7.44 übersichtlich zusammen.

Tabelle 7.3. Arbeitsfaktoren der Lemmata 7.25 bis 7.35, 7.38, 7.42 und 7.44

Lemma	Arbeitsfaktor	Bedingung
7.25	$\alpha(2-\varepsilon,3-\varepsilon)$	$0<\varepsilon\leq 0.545$
7.27	$\alpha(2-\varepsilon,3-\varepsilon)$	$0<\varepsilon<1$
7.30	$\alpha(2-\varepsilon,3-2\varepsilon)$	$0<\varepsilon\leq 0.4$
7.31	$\alpha(1-\varepsilon,5-4\varepsilon)$	$0<\varepsilon<1$
7.33	$\alpha(3-\varepsilon,4-\varepsilon,4-\varepsilon)$	$0<\varepsilon\leq 0.3576$
7.35	$\max\{\alpha(1+\varepsilon,4),\alpha(3,4-\varepsilon,4)\}$	$0<\varepsilon<1$
7.38	$\alpha(4,4,4)$	$0<\varepsilon<1$
7.42	$\alpha(4,4,5,5)$	$0<\varepsilon<1$
7.44	$\alpha(3,3,5)$	$0<\varepsilon\leq 0.287$

Welcher dieser Arbeitsfaktoren am größten ist, hängt von ε ab. Zu beachten ist dabei, dass auch die Arbeitsfaktoren in Tabelle 7.3, in denen ε nicht explizit vorkommt, von ε abhängen. Beispielsweise ergibt sich der Arbeitsfaktor $\alpha(3,3,5)$ in Lemma 7.44 nur, wenn ε im Bereich $0<\varepsilon\leq 0.287$ liegt, wie man im Beweis dieses Lemmas sieht. Dieser Bereich für ε ist der kleinste für alle Arbeitsfaktoren in Tabelle 7.3, und in diesem Bereich sind wir an einem optimalen Wert von ε interessiert.

Für dieses halboffene Intervall $(0, 0.287]$ ermittelten Beigel und Eppstein [BE05] mit Hilfe des Softwarepakets `Mathematica` einen numerischen Wert von $\varepsilon \approx 0.095543$, der den größten der Arbeitsfaktoren in Tabelle 7.3, in denen ε explizit vorkommt, auf ungefähr 1.36443 minimiert. Der Arbeitsfaktor $\alpha(4,4,5,5)$ aus Lemma 7.42, in dem ε nicht explizit vorkommt, ergibt sich ebenfalls zu ungefähr 1.36443. Ist ε nahe bei diesem Wert, dann sind $\alpha(1+\varepsilon,4)$ und $\alpha(3-\varepsilon,4-\varepsilon,4-\varepsilon)$ die zwei größten unter den Arbeitsfaktoren in Tabelle 7.3, die ε explizit enthalten. Da alle anderen Arbeitsfaktoren in Tabelle 7.3, die ε explizit enthalten, unterhalb von 1.36 liegen, muss der wirkliche Wert von ε die Gleichheit

$$\alpha(1+\varepsilon,4) = \alpha(3-\varepsilon,4-\varepsilon,4-\varepsilon) \tag{7.15}$$

erfüllen. Wir zeigen nun, dass dieses wirklich optimale ε sogar

$$\alpha(1+\varepsilon,4) = \alpha(4,4,5,5) = \alpha(3-\varepsilon,4-\varepsilon,4-\varepsilon) \tag{7.16}$$

erfüllt. Dazu betrachten wir die folgende Unterteilung einer Instanz I der Größe n in zunächst zwei kleinere Instanzen I_1 und I_2 mit Arbeitsfaktor $\alpha(1+\varepsilon,4)$, d. h., es gilt $|I_1| = n-(1+\varepsilon)$ und $|I_2| = n-4$. Anschließend unterteilen wir I_1 weiter in drei noch kleinere Instanzen I_{11}, I_{12} und I_{13} mit Arbeitsfaktor $\alpha(3-\varepsilon,4-\varepsilon,4-\varepsilon)$. Da diese Aufteilung von I in die vier Instanzen I_2, I_{11}, I_{12} und I_{13} die beiden Arbeitsfaktoren $\alpha(1+\varepsilon,4)$ und $\alpha(3-\varepsilon,4-\varepsilon,4-\varepsilon)$ kombiniert, muss ihr Arbeitsfaktor dazwischen liegen. Aber wegen (7.15) sind $\alpha(1+\varepsilon,4)$ und $\alpha(3-\varepsilon,4-\varepsilon,4-\varepsilon)$ gleich und somit auch gleich dem Arbeitsfaktor der genannten Aufteilung in vier Instanzen. Diese haben jedoch die folgenden Größen:

$$\begin{aligned}
|I_2| &= n-4,\\
|I_{11}| &= n-(1+\varepsilon)-(3-\varepsilon) = n-4,\\
|I_{12}| &= n-(1+\varepsilon)-(4-\varepsilon) = n-5 \text{ und}\\
|I_{13}| &= n-(1+\varepsilon)-(4-\varepsilon) = n-5,
\end{aligned}$$

was genau dem Arbeitsfaktor $\alpha(4,4,5,5)$ entspricht. Somit ist (7.16) gezeigt, und es ergibt sich die Schranke $\tilde{\mathcal{O}}(\alpha(4,4,5,5)^n) \approx \tilde{\mathcal{O}}(1.36443^n)$ für die Laufzeit des Algorithmus in Abb. 7.30, wenn er auf das Problem $(3,2)$-CSP angesetzt wird.

Wenden wir uns nun der Analyse der Laufzeit dieses Algorithmus für das Problem $(4,2)$-CSP zu, so ist diese am größten, wenn sämtlichen Variablen vier Farben zur Verfügung stehen, d. h., wenn mit $n_3 = 0$ und $n = n_4$ die Größe der Instanz gemäß (7.7) und der Transformation in Abb. 7.17 durch

$$n_3 + (2-\varepsilon)n_4 = (2-\varepsilon)n$$

gegeben ist. Dann hat für $\alpha = \alpha(4,4,5,5) \approx 1.36443$ der Algorithmus in Abb. 7.30 die Laufzeit

$$\tilde{\mathcal{O}}(\alpha^{(2-\varepsilon)n}) \approx \tilde{\mathcal{O}}(\alpha^{(2-0.095543)n}) \approx \tilde{\mathcal{O}}((1.36443^{1.904457})^n) \approx \tilde{\mathcal{O}}(1.80721^n),$$

wenn er auf das Problem $(4,2)$-CSP angesetzt wird. ❑

Satz 7.47 hat eine einfache Folgerung hinsichtlich eines randomisierten Algorithmus für $(d,2)$-CSP, $d > 3$, der Satz 7.22 ergänzt.

Korollar 7.48 (Beigel und Eppstein [BE05]). *Für $d > 3$ kann $(d,2)$-CSP durch einen randomisierten Algorithmus in der erwarteten Zeit $\tilde{\mathcal{O}}((0.4518 \cdot d)^n)$ gelöst werden.*

Beweis. Die Beweisidee besteht darin, für jede Variable zufällig vier Farben zu wählen und dann den Algorithmus aus Abb. 7.30 auf die resultierende $(4,2)$-CSP-Instanz anzuwenden, was für $\varepsilon \approx 0.095543$ pro Versuch die Zeit

$$\tilde{\mathcal{O}}(\alpha(4,4,5,5)^{(2-\varepsilon)n}) \approx \tilde{\mathcal{O}}(1.36443^{(2-\varepsilon)n}) \approx \tilde{\mathcal{O}}(1.8072^n)$$

erfordert. (Es gilt $1.8072 = 4 \cdot 0.4518$.)

Da die Lösbarkeit einer zufällig gewählten $(4,2)$-CSP-Instanz mit Wahrscheinlichkeit $4/d$ erhalten bleibt, ist die erwartete Anzahl der wiederholten Versuche $(d/4)^n$, und es ergibt sich insgesamt die Zeit

$$\tilde{\mathcal{O}}((4 \cdot 0.4518)^n \cdot (d/4)^n) = \tilde{\mathcal{O}}((0.4518 \cdot d)^n),$$

wie gewünscht. ❑

Wie man im Beweis sieht, spielt der Zufall für den Spezialfall $d = 4$ in Korollar 7.48 keine Rolle, denn wenn es sowieso nur vier Farben pro Variable gibt, kann man auch nur genau diese auswählen. Das heißt, wir haben dann eine „erwartete Anzahl“ von nur einem Versuch des Algorithmus aus Abb. 7.30 auszuführen und erhalten genau die Zeitschranke des deterministischen CSP-Algorithmus für $(4,2)$-CSP aus Satz 7.47.

7.5 Anwendung auf Färbbarkeitsprobleme für Graphen

Wie wir bereits aus Abschnitt 7.3 wissen, stellt das Dreifärbbarkeitsproblem einen Spezialfall des Problems $(3,2)$-CSP dar. Deshalb erhalten wir sofort eine weitere Folgerung aus Satz 7.47.

Korollar 7.49 (Beigel und Eppstein [BE05]). *Das Problem* 3-FÄRBBARKEIT *kann in der Zeit $\tilde{\mathcal{O}}(1.36443^n)$ gelöst werden.*

Es geht jedoch noch besser. Durch eine Reihe von Tricks, die zum Teil recht raffiniert und technisch aufwändig sind, kann die Zeitschranke aus Korollar 7.49 weiter gedrückt werden. Den folgenden Satz geben wir ohne Beweis an.

Satz 7.50 (Beigel und Eppstein [BE05]). *Das Problem* 3-FÄRBBARKEIT *kann in der Zeit $\tilde{\mathcal{O}}(1.3289^n)$ gelöst werden.* **ohne Beweis**

Auch für das Kantenfärbbarkeitsproblem kann man mit dieser Methode einen verbesserten Exponentialzeit-Algorithmus erhalten. Die Kantenfärbungszahl $\chi'(G)$ eines Graphen G wurde in Abschnitt 3.4.2 als die kleinste Anzahl von unabhängigen Kantenmengen in G definiert, wobei eine Menge von Kanten in G unabhängig ist, falls keine zwei Kanten in ihr einen gemeinsamen Knoten haben. Das in Abb. 7.29 dargestellte Matching ist ein Beispiel für eine unabhängige Kantenmenge. Wir beschränken uns im Folgenden auf das Problem PARTITION IN DREI UNABHÄNGIGE KANTENMENGEN (siehe Abschnitt 3.4.2), das wir synonym auch als KANTEN-3-FÄRBBARKEIT bezeichnen. Ohne Beschränkung der Allgemeinheit können wir davon ausgehen, dass der maximale Knotengrad des Eingabegraphen drei ist; d. h., es genügt, nur kubische Graphen zu betrachten.

KANTEN-3-FÄRBBARKEIT

Ein erster Ansatz ist, den Fakt auszunutzen, dass jeder Graph mit n Knoten und nicht mehr als drei unabhängigen Kantenmengen höchstens $3n/2$ Kanten haben kann. Dann können wir das entsprechende Kanten-Dreifärbbarkeitsproblem als ein Knoten-Dreifärbbarkeitsproblem für den zugehörigen Kantengraphen[45] (siehe Abschnitt 3.4.2) betrachten, und durch die entsprechende Transformation des gegebenen Graphen in seinen Kantengraphen handeln wir uns höchstens einen Kostenfaktor von $3/2$ ein. Wenden wir nun den Algorithmus aus Satz 7.50 auf den resultierenden Kantengraphen an, so ergibt sich wegen $1.3289^{3n/2} = \left(1.3289^{3/2}\right)^n \approx 1.5319^n$ eine Zeitschranke von $\tilde{\mathcal{O}}(1.5319^n)$.

Doch bevor wir unsere Instanz G des Kanten-Dreifärbbarkeitsproblems in den Kantengraphen $L(G)$ (und somit in eine Instanz des Knoten-Dreifärbbarkeitsproblems) umformen, ist es zweckmäßig, zunächst einige Vereinfachungen in G auszuführen. Das wird uns helfen, eine noch bessere Zeitschranke zu erhalten. Die Hauptidee besteht darin, ein Problem zu betrachten, das sozusagen zwischen dem Kanten- und dem Knoten-Dreifärbbarkeitsproblem liegt. Dieses Zwischenproblem ist wie das Kanten-Dreifärbbarkeitsproblem definiert, nur dass zusätzlich einige Einschränkungen hinzugefügt werden, die verbieten, dass bestimmte Kanten dieselbe Farbe haben. Dieses Problem bezeichnen wir als EINGESCHRÄNKTE KANTEN-3-FÄRBBARKEIT. Kanten, die in einer Instanz dieses Problems einer der hinzugefügten Einschränkungen unterworfen sind, nennen wir *eingeschränkte Kanten*. Für jede solche Einschränkung erhalten wir bei der Transformation einer Instanz G von EINGESCHRÄNKTE KANTEN-3-FÄRBBARKEIT in eine Instanz G' von 3-FÄRBBARKEIT eine zusätzliche Kante im Kantengraph von G, d. h., G' entsteht aus dem Kantengraph $L(G)$, indem diesem die den Einschränkungen von G entsprechenden Kanten hinzugefügt werden. Das Problem EINGESCHRÄNKTE KANTEN-3-FÄRBBARKEIT verallgemeinert das Problem KANTEN-3-FÄRBBARKEIT: Instanzen von KANTEN-3-FÄRBBARKEIT können als Instanzen von EINGESCHRÄNKTE KANTEN-3-FÄRBBARKEIT aufgefasst werden, in denen es keine eingeschränkten Kanten gibt.

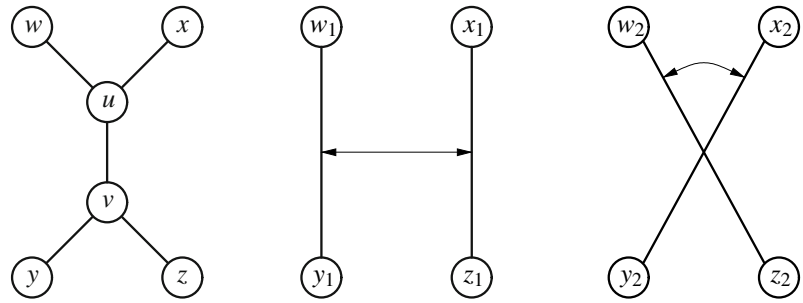
EINGESCHRÄNKTE KANTEN-3-FÄRBBARKEIT

eingeschränkte Kante

Das folgende Lemma gibt an, wie unser Exponentialzeit-Algorithmus für das Problem KANTEN-3-FÄRBBARKEIT (siehe Abb. 7.33) nicht eingeschränkte Kan-

[45] Zur Erinnerung: Der Kantengraph $L(G)$ eines ungerichteten Graphen G besitzt einen Knoten für jede Kante in G, und zwei Knoten sind genau dann in $L(G)$ adjazent, wenn die entsprechenden Kanten in G inzident sind.

ten zwischen zwei Knoten vom Grad drei behandeln wird. Dabei entstehen kleinere Instanzen mit weniger Knoten und weniger Kanten, jedoch kommen dafür Einschränkungen zwischen Kanten hinzu. Dies ermöglicht eine Reduktion, die eine Instanz von KANTEN-3-FÄRBBARKEIT in viele kleinere Instanzen von EINGESCHRÄNKTE KANTEN-3-FÄRBBARKEIT überführt.



(a) Kante $\{u,v\}$ (b) Transformierte Instanz G_1 (c) Transformierte Instanz G_2

Abb. 7.32. Transformation einer uneingeschränkten Kante $\{u,v\}$

Lemma 7.51 (Beigel und Eppstein [BE05]). *Sei G eine Instanz des Problems* EINGESCHRÄNKTE KANTEN-3-FÄRBBARKEIT *mit einer nicht eingeschränkten Kante zwischen zwei Knoten vom Grad drei. Dann kann G durch kleinere Instanzen G_1 und G_2 ersetzt werden, die beide zwei Knoten und drei Kanten weniger als G haben und von denen mindestens eine zu G äquivalent ist.*

Beweis. Abbildung 7.32(a) zeigt die uneingeschränkte Kante $\{u,v\}$ zwischen den Knoten u und v vom Grad drei in einer Instanz G des Problems EINGESCHRÄNKTE KANTEN-3-FÄRBBARKEIT. Die vier Nachbarkanten dieser Kante sind $\{u,w\}$, $\{u,x\}$, $\{v,y\}$ und $\{v,z\}$. Die Kante $\{u,v\}$ kann in einer legalen Kanten-Dreifärbung nur dann gefärbt werden, wenn diese vier inzidenten Kanten insgesamt nur zwei der drei Farben verwenden. Das ist aber nur dann möglich, wenn jeweils zwei unabhängige (also nicht inzidente) Kanten dieser vier Nachbarkanten gleich gefärbt werden: Entweder haben $\{u,w\}$ und $\{v,y\}$ eine Farbe und $\{u,x\}$ und $\{v,z\}$ eine andere Farbe oder es haben $\{u,w\}$ und $\{v,z\}$ eine Farbe und $\{u,x\}$ und $\{v,y\}$ eine andere. Egal welche dieser beiden Färbungen wir verwenden, sie ist als Lösung für G äquivalent zu einer legalen Kanten-Dreifärbung als Lösung für mindestens eine der beiden kleineren Instanzen G_1 oder G_2, die in den Abb.en 7.32(b) und 7.32(c) dargestellt sind. Entscheidend ist nur, dass die Kanten $\{w_1,y_1\}$ und $\{x_1,z_1\}$ in G_1 (bzw. $\{w_2,z_2\}$ und $\{x_2,y_2\}$ in G_2) verschieden gefärbt werden, und das wird durch die als Doppelpfeil in den Abb.en dargestellte Einschränkung zwischen diesen beiden neuen Kanten ausgedrückt. Die Größe dieser Instanzen G_1 und G_2 hat sich gegenüber der Größe von G um zwei Knoten und drei Kanten verringert. ❑

Sei $m = m_3 + m_4$ die Größe einer gegebenen Instanz G des Problems KANTEN-3-FÄRBBARKEIT, wobei m_3 bzw. m_4 die Anzahl der Kanten in G mit drei bzw. vier Nachbarkanten bezeichnet. Kanten mit mehr als vier Nachbarkanten können in kubischen Graphen nicht vorkommen (da sie mindestens einen Endknoten mit Grad mindestens vier hätten und daher nicht legal kanten-dreifärbbar wären), und Kanten mit weniger als drei Nachbarkanten können natürlich kostenlos sofort entfernt werden.

Lemma 7.52 (Beigel und Eppstein [BE05]). *Sei G eine Instanz des Problems* KANTEN-3-FÄRBBARKEIT *mit n Knoten.*

1. *In Polynomialzeit kann*
   a) *entweder eine Menge M von $m_4/3$ Kanten bestimmt werden, sodass Lemma 7.51 unabhängig auf jede Kante in M angewandt werden kann,*
   b) *oder aber festgestellt werden, dass G nicht kanten-dreifärbbar ist.*
2. *Es gilt $6n = 5m_3 + 4m_4$.*

Beweis. Um die erste Aussage zu beweisen, betrachten wir den von den Kanten in G mit vier Nachbarkanten induzierten Teilgraphen G' von G. Offenbar hat G' genau m_4 Kanten. Ist G' kanten-dreifärbbar, so partitioniert deshalb jede legale Kanten-Dreifärbung die Kantenmenge von G' in drei Teilmengen, von denen mindestens eine wenigstens $m_4/3$ Kanten enthält. Ein maximales Matching von G' hat also mindestens diese Größe. Auch wenn wir in Polynomialzeit natürlich nicht entscheiden können, ob G' kanten-dreifärbbar ist (da dieses Problem – sogar für kubische Graphen – NP-vollständig ist [Hol81]), so lässt sich der Spieß doch wie folgt herumdrehen: Findet ein Matching-Algorithmus (siehe Kapitel 3) *kein* maximales Matching der Größe mindestens $m_4/3$ in G', so kann G' nicht kanten-dreifärbbar sein. Dann ist aber auch G keine Ja-Instanz von KANTEN-3-FÄRBBARKEIT, und wir können den Algorithmus in Abb. 7.33 sofort erfolglos terminieren lassen. Findet ein Matching-Algorithmus jedoch ein maximales Matching der Größe mindestens $m_4/3$ in G', so haben wir eine Menge M der Größe $m_4/3$ in Polynomialzeit gefunden, sodass Lemma 7.51 unabhängig auf jede Kante in M angewandt werden kann. Die Anwendung von Lemma 7.51 auf irgendeine Kante in M schränkt weder irgendeine andere Kante in M ein noch führt sie dazu, dass die restlichen Kanten in M kein Matching mehr bilden.

Um die zweite Aussage des Lemmas zu zeigen, weisen wir den Knoten und Kanten des Graphen G die folgenden Kosten zu:

1. Jeder Knoten von G hat die Kosten $6/5$, die dann gleichmäßig auf alle seine inzidenten Kanten verteilt werden.
2. Zusätzlich erhält jede Kante mit vier Nachbarkanten die Kosten von $1/5$.

Demnach erhält eine Kante zwischen zwei Knoten vom Grad drei von ihren beiden Knoten jeweils zwei Fünftel der Kosteneinheit und trägt selbst ein weiteres Fünftel bei, was sich zu einer Kosteneinheit summiert. Eine Kante zwischen einem Knoten vom Grad drei und einem Knoten vom Grad zwei erhält zwei Fünftel der Kosteneinheit vom einen Knoten und drei Fünftel vom anderen, trägt selbst aber nichts bei,

weshalb sie in der Summe wieder auf eine Kosteneinheit kommt. Da wir Kanten mit weniger als drei Nachbarkanten bereits entfernt haben, gibt es keine Kante zwischen zwei Knoten vom Grad höchstens zwei. (Wie bereits erwähnt, können Kanten mit mehr als vier Nachbarkanten in kubischen Graphen nicht vorkommen.) Da jede Kante Einheitskosten hat, ergeben sich für den Graphen G insgesamt die Kosten von

$$m_3 + m_4 = m. \tag{7.17}$$

Zählen wir die Kosten für G jedoch etwas anders zusammen, so kommen wir auf

$$\frac{6n}{5} + \frac{m_4}{5}, \tag{7.18}$$

da jeder der n Knoten Kosten von $6/5$ und jede der m_4 Kanten mit vier Nachbarkanten Kosten von $1/5$ beiträgt. Natürlich ist es egal, wie wir die Kosten für G ermitteln; die beiden Werte in (7.17) und (7.18) müssen gleich sein. Somit ergibt sich:

$$\frac{6n}{5} + \frac{m_4}{5} = m = m_3 + m_4,$$

und wenn wir auf beiden Seiten dieser Gleichung mit 5 multiplizieren und m_4 abziehen, erhalten wir wie gewünscht die Gleichung $6n = 5m_3 + 4m_4$. ❑

Abbildung 7.33 zeigt einen Algorithmus für das Problem KANTEN-3-FÄRBBARKEIT.

Satz 7.53 (Beigel und Eppstein [BE05]). *Sei G ein Graph mit n Knoten. In der Zeit*

$$\tilde{\mathcal{O}}(1.4143^n)$$

kann der in Abb. 7.33 dargestellte Algorithmus entweder eine legale Kanten-Dreifärbung für G finden oder feststellen, dass G nicht kanten-dreifärbbar ist.

Beweis. Nach der ersten Aussage von Lemma 7.52 können wir in Polynomialzeit entweder eine Menge M von $m_4/3$ Kanten bestimmen, sodass Lemma 7.51 unabhängig auf jede Kante in M angewandt werden kann, oder aber feststellen, dass G nicht kanten-dreifärbbar ist. Angenommen, G ist kanten-dreifärbbar und wir haben eine solche Menge M bestimmt. Für jede der $m_4/3$ Kanten in M gibt es zwei Möglichkeiten, sie in G zu ersetzen, entweder gemäß Abb. 7.32(b) oder gemäß Abb. 7.32(c). Somit erhalten wir $\ell = 2^{m_4/3}$ kleinere Instanzen $G_1, G_2, \ldots, G_\ell$ mit eingeschränkten Kanten, von denen jede nur m_3 Kanten hat. Diese Instanzen des Problems EINGESCHRÄNKTE KANTEN-3-FÄRBBARKEIT formen wir nun in ihre Kantengraphen um und fügen diesen für ihre Einschränkungen Kanten hinzu; wie im Algorithmus in Abb. 7.33 beschrieben, erhalten wir so einen Graphen G'_i aus G_i für jedes i, $1 \leq i \leq \ell$.

Auf die resultierenden Graphen $G'_1, G'_2, \ldots, G'_\ell$, die wir als Instanzen von 3-FÄRBBARKEIT auffassen, wenden wir schließlich den Algorithmus aus Satz 7.50 an, der in der Zeit $\tilde{\mathcal{O}}(1.3289^n)$ läuft. Da jeder der Graphen G'_i höchstens m_3 Knoten hat, lässt sich die Laufzeit des Algorithmus für KANTEN-3-FÄRBBARKEIT aus Abb. 7.33 insgesamt also durch

1. Bestimme den Teilgraphen G' von G, der von den Kanten in G mit vier Nachbarkanten induziert wird, und finde mit einem Matching-Algorithmus ein maximales Matching M' in G'. Enthält M' weniger als $m_4/3$ Kanten, so gib die Meldung „G ist nicht kanten-dreifärbbar“ aus und terminiere erfolglos (siehe den ersten Teil von Lemma 7.52); andernfalls sei M eine Teilmenge von M' mit $|M| = m_4/3$.
2. Wende Lemma 7.51 auf jede Kante in M an. Wie im Beweis von Satz 7.53 beschrieben, kann G so durch $\ell = 2^{m_4/3}$ kleinere Instanzen $G_1, G_2, \ldots, G_\ell$ mit eingeschränkten Kanten ersetzt werden, die jeweils nur m_3 Kanten haben. Bilde für jedes i, $1 \leq i \leq \ell$ aus G_i einen Kantengraphen und füge diesem für die Einschränkungen in G_i wie folgt Kanten hinzu:
   a) Für jede Kante in G_i hat G_i' einen Knoten;
   b) zwei Knoten in G_i' werden genau dann durch eine Kante verbunden, wenn die entsprechenden Kanten in G_i einen gemeinsamen Knoten haben;
   c) zusätzlich wird für jede Einschränkung zwischen zwei Kanten e_j und e_k in G_i eine Kante in G_i' zwischen den e_j und e_k entsprechenden Knoten hinzugefügt. (Beachte, dass es in G_i nur zwischen unabhängigen Kanten Einschränkungen gibt.)
3. Wende den Algorithmus aus Satz 7.50 auf die resultierenden Instanzen $G_1', G_2', \ldots, G_\ell'$ von 3-FÄRBBARKEIT an.

Abb. 7.33. Algorithmus für KANTEN-3-FÄRBBARKEIT.

$$\tilde{\mathcal{O}}(1.3289^{m_3} \cdot 2^{m_4/3}) \tag{7.19}$$

abschätzen. Nach der zweiten Aussage von Lemma 7.52 wissen wir, dass

$$m_3 = \frac{6n}{5} - \frac{4m_4}{5}$$

gilt; folglich lässt sich (7.19) als

$$\tilde{\mathcal{O}}\left(1.3289^{6n/5} \cdot \left(2^{1/3} \cdot 1.3289^{-4/5}\right)^{m_4}\right)$$

schreiben. Da $2^{1/3} \cdot 1.3289^{-4/5} > 1$ ist, wird diese Schranke dann am größten, wenn m_4 größtmöglich ist, was genau dann der Fall ist, wenn $m_4 = 3n/2$ und $m_3 = 0$ gilt. In diesem Fall erhalten wir aber aus (7.19) die Schranke

$$\tilde{\mathcal{O}}\left(1.3289^0 \cdot 2^{(3n/2)/3}\right) = \tilde{\mathcal{O}}\left(2^{n/2}\right) = \tilde{\mathcal{O}}\left(\left(2^{1/2}\right)^n\right), \tag{7.20}$$

und wegen $2^{1/2} < 1.4143$ ergibt sich die im Lemma genannte Laufzeit von

$$\tilde{\mathcal{O}}(1.4143^n),$$

womit der Beweis abgeschlossen ist. ❑

7.6 Literaturhinweise

Exakte Exponentialzeit-Algorithmen für schwere Probleme werden seit einigen Jahrzehnten – und besonders intensiv in den letzten Jahren – untersucht. Unter der vernünftigen Annahme, dass $P \neq NP$ gilt, ist für keines dieser Probleme ein Polynomialzeit-Algorithmus bekannt. Das Nichtvorhandensein effizienter Algorithmen lässt die Probleme jedoch nicht einfach verschwinden. Sie sind wichtig, und man sollte deshalb keine Mühe scheuen, die vorhandenen Algorithmen weiter zu verbessern, um diese Probleme *so schnell wie möglich* lösen zu können, oder andere Möglichkeiten zu suchen, ihnen beizukommen.

Neben dem Entwurf von effizienten, jedoch nicht immer korrekten Heuristiken, von Algorithmen mit einem guten Average-case-Verhalten und von Approximationsalgorithmen (drei Ansätze, die hier nicht behandelt werden) sowie von Fest-Parameter-Algorithmen (siehe Kapitel 6) besteht ein wichtiger Ansatz für den Umgang mit schweren Problemen in der Verbesserung der *exakten Algorithmen* für diese Probleme in der *Worst-case-Analyse*. Wie in Abschnitt 7.1.5 bereits erläutert wurde, ist dies ein Ansatz, der gerade unter praktischem Aspekt von großer Bedeutung ist: Die Probleme sind für Eingaben moderater Größe in vernünftiger Zeit lösbar, auch wenn wir uns asymptotisch – für sehr große Eingaben – immer noch mit den Nachteilen der Exponentialzeit plagen müssen.

Dieser Ansatz wurde für viele Probleme recht erfolgreich verfolgt, sowohl für schwere Graphenprobleme wie etwa UNABHÄNGIGE MENGE, DOMINIERENDE MENGE, DOMATISCHE ZAHL, das TRAVELING SALESPERSON PROBLEM und verschiedene Graphfärbbarkeitsprobleme als auch für schwere Probleme aus anderen Bereichen wie das Erfüllbarkeitsproblem der Aussagenlogik, SAT, und allgemeiner das Constraint Satisfaction Problem. Wir behandeln diese Probleme hier nicht im Detail, erwähnen aber, dass es beispielsweise Algorithmen gibt, die unabhängige Mengen in einem Graphen in moderater Exponentialzeit finden (siehe z. B. [TT77, Rob86]); der derzeit beste dieser Algorithmen geht auf Robson [Rob01] zurück und erreicht die Schranke $\tilde{\mathcal{O}}(1.1892^n)$. Für eine Vielzahl weiterer Ergebnisse verweisen wir auf die Übersichtsartikel von Woeginger [Woe03], Schöning [Sch05] und Riege und Rothe [RR06].

Für schwere Graphfärbbarkeitsprobleme wurden exakte Exponentialzeit-Algorithmen in diesem Kapitel vorgestellt. Einerseits haben wir dabei ein ganz frühes solches Resultat herausgegriffen, nämlich den Algorithmus von Lawler [Law76] zur Berechnung der Färbungszahl eines gegebenen Graphen in der Zeit $\tilde{\mathcal{O}}(2.4423^n)$ (Satz 7.1),[46] der sich zu einem $\tilde{\mathcal{O}}(1.4423^n)$-Algorithmus für das Problem 3-FÄRBBARKEIT modifizieren lässt (Satz 7.3). Andererseits haben wir in den Abschnitten 7.3 bis 7.5 die Methode von Beigel und Eppstein [BE05] (siehe auch die früheren Arbeiten [BE95, Epp01]) ausführlich dargestellt, mit der sich

1. 3-FÄRBBARKEIT, aufgefasst als ein Spezialfall von $(3,2)$-CSP, durch einen einfachen randomisierten Algorithmus in der erwarteten Zeit $\tilde{\mathcal{O}}(1.4143^n)$ (siehe

[46] Eppstein [Epp03] verbesserte die Zeitschranke für die Berechnung der chromatischen Zahl zu $\tilde{\mathcal{O}}(2.4150^n)$ und Byskov [Bys02] weiter zu $\tilde{\mathcal{O}}(2.4023^n)$.

Satz 7.22) und deterministisch – unter der Verwendung von Eppsteins Methode der quasi-konvexen Programmierung [Epp04] – sogar in der Zeit $\tilde{\mathcal{O}}(1.36443^n)$ lösen lässt (siehe Korollar 7.49),
2. durch einigen weiteren technischen Aufwand sogar die derzeit beste Schranke von $\tilde{\mathcal{O}}(1.3289^n)$ für 3-FÄRBBARKEIT zeigen lässt (siehe Satz 7.50) und
3. KANTEN-3-FÄRBBARKEIT in der Zeit $\tilde{\mathcal{O}}(1.4143^n)$ lösen lässt (siehe Satz 7.53).

Natürlich liegen zwischen diesen beiden Meilensteinen, dem Resultat von Lawler und dem von Beigel und Eppstein, viele schrittweise Verbesserungen, die sich zahlreicher, ganz unterschiedlicher Ideen bedienen. So gelang beispielsweise Schiermeyer [Sch93] mit einem recht komplizierten Algorithmus und einer tiefschürfenden Analyse der Nachweis einer Schranke von $\tilde{\mathcal{O}}(1.415^n)$ für 3-FÄRBBARKEIT, die er später noch bis auf $\tilde{\mathcal{O}}(1.3977^n)$ drücken konnte [Sch96].

Die Methode von Beigel und Eppstein [BE05] fasst Färbbarkeitsprobleme für Graphen im Sinne von *Constraint Satisfaction* auf. Es ist nicht verwunderlich, dass in ähnlicher erfolgreicher Weise ein solcher CSP-Zugang auch für andere Probleme gelungen ist, beispielsweise bei Schönings randomisiertem Exponentialzeit-Algorithmus für k-SAT [Sch99].

Abschließend erwähnen wir noch einige weitere Resultate zu Graphfärbungsproblemen. Blum und Karger [BK97] bewiesen, dass in Polynomialzeit jeder dreifärbbare Graph mit $\tilde{\mathcal{O}}(n^{3/14})$ Farben gefärbt werden kann. Alon and Kahale [AK97] zeigten, wie man mit der so genannten Spektraltechnik dreifärbbare Zufallsgraphen in erwarteter Polynomialzeit legal mit drei Farben färben kann. Petford und Welsh [PW89] beschäftigten sich ebenfalls mit dreifärbbaren Zufallsgraphen. Sie entwarfen einen randomisierten Algorithmus, der empirischen Daten zufolge für zufällig gewählte dreifärbbare Graphen eine gute Laufzeit hat, auch wenn sie keine Schranken für die Laufzeit bewiesen. Vlasie [Vla95] zeigte, wie man systematisch eine Klasse von Graphen erzeugen kann, die – anders als zufällig gewählte dreifärbbare Graphen – schwer zu färben sind. Khanna, Linial und Safra [KLS00] untersuchten die Frage, wie schwer es ist, die chromatische Zahl eines Graphen zu approximieren. Motiviert durch diese Ergebnisse zeigten Guruswami und Khanna [GK04], dass es NP-hart ist, einen dreifärbbaren Graphen mit vier Farben zu färben. Dieses Resultat und sein Beweis wiederum waren der Schlüssel für die Lösung eines offenen Problems von Wagner [Wag87]: Rothe [Rot03] klassifizierte die Komplexität des exakten Vierfärbbarkeitsproblems, das für einen gegebenen Graphen G fragt, ob $\chi(G) = 4$ gilt. Hinsichtlich weiterer komplexitätstheoretischer Ergebnisse für Graphfärbbarkeitsprobleme verweisen wir auf [CM87, Rot00] und die darin zitierte Literatur.