

RHEINISCH-WESTFÄLISCHE TECHNISCHE HOCHSCHULE AACHEN

LEHRSTUHL FÜR HOCHLEISTUNGSRECHNEN

Prof. Christian Bischof, Ph.D.

**Graphfärbung
zur partiellen Berechnung
von Jacobi-Matrizen**

Diplomarbeit

von
cand. Inform. Michael Lülkesmann
Matr.-Nr. 235946

29. September 2006

Erstgutachter: Priv.-Doz. Dr.-Ing. Martin Bucker
Zweitgutachter: Prof. Dr. Peter Rossmanith

Betreuer: Dipl.-Inf. Andreas Wolf

Erklärung

Hiermit versichere ich, dass ich die Arbeit selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt sowie Zitate kenntlich gemacht habe.

Aachen, im September 2006

Danksagung

Zunächst möchte ich mich bei Herrn Priv.-Doz. Dr.-Ing. Martin Bucker vom Lehrstuhl für Hochleistungsrechnen für die Vergabe des interessanten Themas und die sehr gute Betreuung und bei Herrn Prof. Dr. Peter Rossmanith vom Lehr- und Forschungsgebiet Informatik I für sein Zweitgutachten bedanken.

Des Weiteren möchte ich mich auch bei Herrn Dipl.-Inform. Andreas Wolf für die sehr gute Betreuung und bei allen anderen wissenschaftlichen Mitarbeitern vom Lehrstuhl für Hochleistungsrechnen, die mir mit Rat zur Seite standen, bedanken.

Abschließend möchte ich mich bei meinen Eltern für das Ermöglichen des Studiums und bei meiner Freundin Christina Strob für die Unterstützung und das Korrekturlesen bedanken.

Inhaltsverzeichnis

1	Einleitung	1
2	Motivation	3
2.1	Ableitungsberechnung	4
2.1.1	Symbolisches Differenzieren	4
2.1.2	Finite Differenzen	5
2.1.3	Automatisches Differenzieren	5
2.2	Vorkonditionierung	9
2.3	Techniken zum Seeden	10
2.4	Einführung in die Graphfärbung	15
2.5	Partielle Berechnung von Jacobi-Matrizen	18
3	Färbungsprobleme	21
3.1	Vollständige Färbungen	23
3.1.1	Einseitige Färbung	23
3.1.2	Zweiseitige Färbung	25
3.2	Beschränkte Färbungen	29
3.2.1	Beschränkte einseitige Färbung	30
3.2.2	Beschränkte zweiseitige Färbung	32
3.2.3	Färbungen zur partiellen Berechnung der Hauptdiagonalelemente	36
4	Färbungsheuristiken	43
4.1	Einseitige Färbungen	43
4.1.1	Vollständige Färbung	44
4.1.2	Beschränkte Färbung	45
4.1.3	Vorsortierung	47
4.2	Zweiseitige Färbungen	50
4.2.1	Vollständige zweiseitige Färbung	50
4.2.2	Beschränkte zweiseitige Färbung	52
4.2.3	Vorsortierung	56

5	Ergebnisse und Bewertung	59
5.1	Vollständige Färbungen	60
5.2	Beschränkte Färbung	65
6	Zusammenfassung und Ausblick	71
	Literaturverzeichnis	73

1 Einleitung

Im wissenschaftlichen Rechnen sind häufig kombinatorische Probleme zu lösen, die durch Graphen modelliert werden können. Insbesondere sind Graphen allgegenwärtig in der numerischen linearen Algebra, wenn die zugrundeliegenden Matrizen dünnbesetzt sind. Der Ursprung der in dieser Arbeit betrachteten Problemstellungen liegt in der Lösung eines linearen Gleichungssystems, dessen Koeffizientenmatrix eine dünnbesetzte, große und nicht-singuläre Jacobi-Matrix J ist. Diese Systeme werden mit iterativen Verfahren gelöst, die in der Praxis durch eine Vorkonditionierung beschleunigt werden. Zur Vorkonditionierung können Methoden verwendet werden, die nur eine Teilmenge der Nichtnullelemente von J benötigen. Bisher wurden zur Bestimmung des Vorkonditionierers alle Elemente von J berechnet. Darum wird dieses Vorgehen als vollständige Berechnung bezeichnet. Bei der partiellen Berechnung muss hingegen nur noch die Teilmenge der Elemente, die zur Vorkonditionierung benötigt werden, mit möglichst geringem Aufwand berechnet werden.

Bisher wurde eine Jacobi-Matrix J vollständig berechnet, z.B. mit den Finiten Differenzen oder dem Automatischen Differenzieren. Mit entsprechender Initialisierung können dabei Berechnungen gespart werden, da mehrere Zeilen/Spalten von J durch eine einzelne Linearkombination berechnet werden können. Die Elemente aus J sollen dabei aus den Linearkombinationen abgelesen werden können. Linearkombinationen nur als Spalten oder nur als Zeilen können durch eine sogenannte unidirektionale Partitionierung von J berechnet werden. Bei der bidirektionalen Partitionierung werden Linearkombinationen als Spalten und Zeilen berechnet. Zur partiellen Berechnung wird die Initialisierung so angepasst, dass J oftmals mit noch weniger Linearkombinationen partiell berechnet werden kann.

Die unidirektionale Partitionierung von J kann als einseitige Graphfärbung modelliert werden und die bidirektionale Partitionierung als zweiseitige Färbung. Hierzu wird die Dünnbesetztheitsstruktur einer Jacobi-Matrix J durch einen bipartiten Graphen eineindeutig dargestellt. So ein Graph kann dann entsprechend gefärbt werden. Die Spalten bzw. Zeilen, deren entsprechende Knoten im Graphen identisch gefärbt sind, können durch eine einzelne Linearkombination berechnet werden. Die Färbung, die zu einer partiellen Berechnung von J führt, wird beschränkt genannt, da sie nur eine Teilmenge der Elemente von J berücksichtigt. Aus den Färbungen, die der vollständigen Berechnung genügen, kann abgeleitet werden, welche Spalten bzw. Zeilen von J als eine Linearkombination berechnet werden können. Darum wird in diesem Fall auch von vollständiger Färbung gesprochen, da alle Elemente von J berechnet werden sollen.

Die Elemente von J können bestimmt werden, indem nur Spalten, nur Zeilen (unidirekti-

onale Partitionierung, einseitige Färbung) oder eine Kombination aus Spalten und Zeilen (bidirektionale Partitionierung, zweiseitige Färbung) berechnet werden. Daraus ergeben sich vier Graphfärbungsprobleme: einseitige und vollständige, zweiseitige und vollständige, einseitige und beschränkte sowie zweiseitige und beschränkte Färbung.

In der vorliegenden Arbeit werden für diese vier Probleme jeweils ein Minimierungs- und ein Entscheidungsproblem definiert. Danach wird für jedes Entscheidungsproblem ein NP-Vollständigkeitsbeweis geführt, womit gezeigt wird, dass die zugehörigen Minimierungsprobleme NP-schwierig sind. Zudem wird ein Spezialfall der beschränkten Färbung betrachtet, bei dem nur die Hauptdiagonalelemente benötigt werden, weil nur diese Elemente für eine diagonale Vorkonditionierung benötigt werden. Es wird dafür gezeigt, dass die einseitige Färbung und die zweiseitige Färbung immer dieselbe minimale Farbanzahl für einen Graphen benötigen. Für diesen Spezialfall wird zusätzlich gezeigt, dass das Problem der einseitigen Färbung NP-vollständig ist.

Nachdem bewiesen worden ist, dass diese Probleme NP-schwierig sind, werden Heuristiken zum Färben der bipartiten Graphen vorgestellt. Dazu wird eine Heuristik zur einseitigen und vollständigen Färbung und eine Heuristik zur zweiseitigen und vollständigen Färbung betrachtet. In der vorliegenden Arbeit werden diese erstmals auf die jeweilige partielle Färbung angepasst und anschließend implementiert.

Zudem werden Algorithmen zur Vorsortierung einer Knotenmenge betrachtet, um die Anzahl der benötigten Farben bei der einseitigen und vollständigen Färbung zu reduzieren. Die Algorithmen müssen hier zuerst auf den bipartiten Graphen angepasst werden, weil sie für eine andere Graphklasse entwickelt wurden. Im zweiten Schritt werden diese für die einseitige und partielle Färbung angepasst. Schließlich wird gezeigt, dass das Anwenden der Vorsortierungen vor den zweiseitigen Färbungen auch oft zu einer Reduzierung der Farbanzahl führt.

Dann werden die bipartiten Graphen verschiedener Testmatrizen zuerst vollständig und dann beschränkt gefärbt, wobei bei der beschränkten Färbung die Hauptdiagonalelemente die benötigten Elemente sind. Zusätzlich werden die bipartiten Graphen gefärbt, nachdem die Knoten vorsortiert worden sind. Abschließend werden die Ergebnisse verglichen und ausgewertet.

Die vorliegende Arbeit ist wie folgt aufgebaut: In Kapitel 2 wird die partielle Berechnung der Jacobi-Matrizen und die Modellierung als Graphfärbungsproblem motiviert. In Kapitel 3 werden die Graphfärbungen und die entsprechenden Probleme definiert und die NP-Vollständigkeitsbeweise geführt. Nachdem die NP-Vollständigkeit für die Probleme gezeigt wurde, werden in Kapitel 4 existierende Färbungs- und Vorsortierungsheuristiken erläutert und neue Algorithmen entwickelt. In Kapitel 5 werden Messungen durchgeführt und die Ergebnisse ausgewertet. Die Arbeit wird mit einer Zusammenfassung und einem Ausblick abgeschlossen.

2 Motivation

Im wissenschaftlichen Rechnen müssen oftmals lineare Gleichungssysteme der Form

$$Az = b \tag{2.1}$$

gelöst werden, wobei $A \in \mathbb{R}^{n \times n}$, groß, regulär und dünnbesetzt ist, und $z, b \in \mathbb{R}^n$ sind. Solche Systeme kommen zum Beispiel beim numerischen Lösen von partiellen Differentialgleichungen vor, die im Zusammenhang mit der Finite-Elemente-Methode (FEM) in Anwendungsbereichen wie etwa der Numerischen Strömungsmechanik (engl. computational fluid dynamics, CFD) auftreten.

Das lineare Gleichungssystem (2.1) kann entweder durch ein direktes oder durch ein iteratives Verfahren gelöst werden. Oftmals wird ein iteratives Verfahren gewählt, da mit diesem eine approximative Lösung, die hinreichend genau ist, schneller berechnet werden kann, als eine exakte Lösung mit einer direkten Methode. Direkte Verfahren liefern immer exakte Lösungen, benötigen aber mehr Operationen und mehr Speicherplatz.

Nach der *Vorkonditionierung* eines linearen Gleichungssystems kann die Lösung durch ein iteratives Lösungsverfahren oftmals noch schneller berechnet werden. Hierzu wird eine Matrix M , der Vorkonditionierer, bestimmt. Bei der Links-Vorkonditionierung erhält man z.B. folgendes Gleichungssystem:

$$M^{-1}Az = M^{-1}b.$$

Ein Vorkonditionierer M kann aus einer Teilmenge der Elemente von A bestimmt werden. Um den Vorkonditionierer M zu erhalten, muss die Matrix A nicht vollständig berechnet werden, sondern es reicht aus, nur die benötigten Elemente zu bestimmen.

Die Matrix A kann eine Jacobi-Matrix $J|_{x_0}$ an einem Punkt $x_0 \in \mathbb{R}^n$ sein – wie z.B. beim Newton-Verfahren. Hierbei kann die Funktion f , deren 1. Ableitung J ist, als numerisches (Simulations-)Programm gegeben sein. Zur Berechnung einer solchen Jacobi-Matrix existiert neben dem *Symbolischen Differenzieren* auch das Verfahren der *Finiten Differenzen* und die Technik des *Automatischen Differenzierens*.

Beim *Automatischen Differenzieren* (AD) wird die Originalfunktion in eine Funktion transformiert, mit der eine oder mehrere Richtungsableitungen (Spalten der Jacobi-Matrix) bzw. Adjungierte (Zeilen der Jacobi-Matrix) berechnet werden können. Meistens wird dabei auch das Ergebnis der Originalfunktion zurückgegeben.

Unterschiedliche *Techniken zum Seeden* ermöglichen es, eine Linearkombination von mehreren Richtungsableitungen bzw. Adjungierten zu berechnen. Also wird, z.B. in einem AD-Berechnungsdurchlauf, nicht nur eine, sondern mehrere Richtungsableitungen berechnet. Dieses führt zu einer Reduzierung der Kosten, da weniger AD-Berechnungsdurchläufe benötigt werden. Nun muss eine Seed-Matrix ermittelt werden, die angibt, welche Spalten bzw. Zeilen der Jacobi-Matrix gleichzeitig berechnet werden dürfen, so dass alle Elemente der Jacobi-Matrix direkt aus dem Ergebnis bestimmt werden können. Vor allem bei dünnbesetzten Matrizen führt dieses Zusammenfassen in vielen Fällen zu einer erheblichen Reduzierung des Berechnungsaufwands. Wenn nicht alle Elemente der Jacobi-Matrix J benötigt werden, dann ist oftmals eine weitere Reduzierung der AD-Berechnungsdurchläufe möglich.

Die einzelnen Punkte dieser Übersicht werden nun in den folgenden Abschnitten genauer erläutert.

2.1 Ableitungsberechnung

Gegeben sei die numerische (Simulations-)Funktion

$$f : \mathbb{R}^n \rightarrow \mathbb{R}^m, \quad y = f(x). \quad (2.2)$$

Die erste Ableitung dieser Funktion an dem Punkt x_0 soll berechnet werden, so dass wir die Jacobi-Matrix

$$J(x)|_{x=x_0} = J|_{x_0} = f'(x)|_{x=x_0} \equiv \left(\frac{\delta y_j}{\delta x_i} \right) \bigg|_{x=x_0}^{j=1,\dots,m, i=1,\dots,n} \quad (2.3)$$

erhalten, wobei $J \in \mathbb{R}^{m \times n}$ ist. Für das Differenzieren gibt es unterschiedliche Verfahren, von denen drei im Folgenden vorgestellt werden: das *Symbolische Differenzieren*, die *Finiten Differenzen* (FD) und das *Automatische Differenzieren* (AD).

2.1.1 Symbolisches Differenzieren

Beim *Symbolischen Differenzieren* wird die Jacobi-Matrix

$$J = \left(\frac{\delta y_j}{\delta x_i} \right) \bigg|_{i=1,\dots,n}^{j=1,\dots,m}$$

als eine explizite Formel berechnet. Die Matrix J kann mit einem Computeralgebra-System wie Maple oder Mathematica berechnet werden. Um $J|_{x_0}$ zu erhalten, müssen die Werte eines Punktes x_0 noch in die berechnete Jacobi-Matrix J eingesetzt werden.

Nun ist aber keine explizite Formel, sondern eine numerische Funktion gegeben, deren 1. Ableitung berechnet werden soll. Aus verschiedenen Gründen kann dieses Verfahren nicht

sinnvoll angewendet werden. Zum einen müssten alle Schleifen, bedingte Verzweigungen und Unterprogrammaufrufe in explizite Formeln transformiert werden. Bei großen Funktionen ist der Aufwand dafür hoch. Zum anderen verdoppelt sich die Darstellung der expliziten Formel mit jeder nichtlinearen Operation, wie z.B. $*$ und $/$.

2.1.2 Finite Differenzen

Durch das FD-Verfahren [22] kann eine Richtungsableitung (Gradient an der Stelle x_0) einer numerischen Funktion approximativ berechnet werden. Alle Richtungsableitungen zusammen ergeben die Jacobi-Matrix (2.3) der Funktion (2.2). Es gibt die Vorwärts-, die Rückwärtsdifferenzen und die zentrierten Differenzen. Bei den zentrierten Differenzen, die eine vergleichsweise hohe Genauigkeit aufweisen, wird die k -te Richtungsableitung der Jacobi-Matrix einer Funktion f wie folgt bestimmt:

$$\frac{\delta}{\delta x_k} f(x) \cong \frac{1}{h} [f(x + \frac{h}{2} e_k) - f(x - \frac{h}{2} e_k)], \quad 1 \leq k \leq n,$$

wobei durch den Parameter h die Schrittweite festgelegt wird und e_k der k -te kanonische Einheitsvektor ist.

Die FD lassen sich zwar einfach berechnen, da die Funktion nur zweimal mit unterschiedlichen Eingabewerten aufgerufen und anschließend nur die skalierte Differenz gebildet werden muss. Allerdings gibt es Schwierigkeiten bei der Bestimmung des Wertes von h , auf die in [14, Kap. 1] eingegangen wird. Zum einen soll der Abbruchfehler möglichst klein sein, so dass h klein gewählt werden muss (Stichwort: Taylorreihe). Zum anderen soll der Auslöschungsfehler möglichst klein sein, so dass h groß sein muss, damit keine Nachkommastellen wegen der endlichen Rechengenauigkeit ausgelöscht werden.

2.1.3 Automatisches Differenzieren

Eine Funktion (2.2) kann durch das *Automatische Differenzieren* (AD) [14, 20] in eine Funktion transformiert werden, mit der die 1. Ableitung der Originalfunktion berechnet werden kann. Der größte Vorteil gegenüber den FD ist die Tatsache, dass weder der Abbruchfehler noch der Auslöschungsfehler der FD auftreten.

Beim AD gibt es zwei unterschiedliche Verfahren: zum einen den Vorwärtsmodus (engl. Forward Mode (FM)) und zum anderen den Rückwärtsmodus (engl. Reverse Mode (RM)). Von jetzt an bezeichnet J die Jacobi-Matrix $J|_{x=x_0}$.

Beim *Vorwärtsmodus* kann die Originalfunktion (2.2) zu einer Funktion transformiert werden, die eine Richtungsableitung

$$\dot{f} : \mathbb{R}^{2n} \rightarrow \mathbb{R}^m : \dot{y} = \dot{f}(x, \dot{x}) = f'(x) \cdot \dot{x}, \quad (2.4)$$

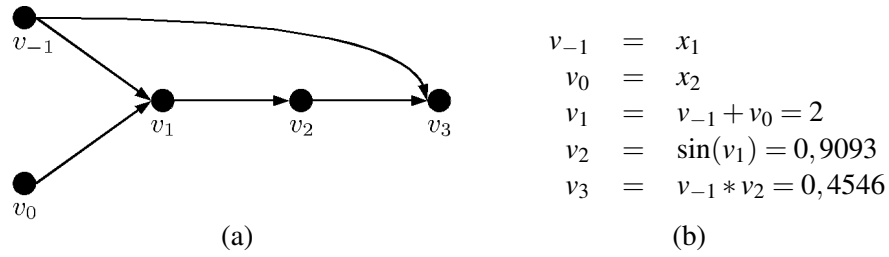


Abbildung 2.1.1: (a) Berechnungsgraph zur Funktion $f(x_1, x_2) = x_1 * \sin(x_1 + x_2)$
(b) Entsprechende Berechnungsschritte für $v_{-1} = 0,5$ und $v_0 = 1,5$

berechnet, wobei die Richtung $\dot{x} \in \mathbb{R}^n$ genau einmal mit jedem kanonischen Einheitsvektor $e_i = (0, \dots, 1, \dots, 0)^T$, $i = 1, \dots, n$, initialisiert werden muss, um die Jacobi-Matrix J zu ermitteln. Die Vektoren \dot{x} werden auch Seed-Vektoren oder Richtungsvektoren genannt, da sie die Richtung angeben, zu welcher eine Ableitung berechnet werden soll. Zu einem Richtungsvektor $\dot{x} = e_i$ wird eine Richtungsableitung $\dot{y} = (\frac{\delta y_1}{\delta x_i}, \dots, \frac{\delta y_m}{\delta x_i})^T$ berechnet, die auch als i -te Spalte von J betrachtet werden kann.

In der transformierten Funktion gibt es die Eingabevariablen x_1, \dots, x_n (unabhängige Variablen), die Zwischenvariablen v_1, \dots, v_q und die Ausgabevariablen y_1, \dots, y_m (abhängige Variablen).

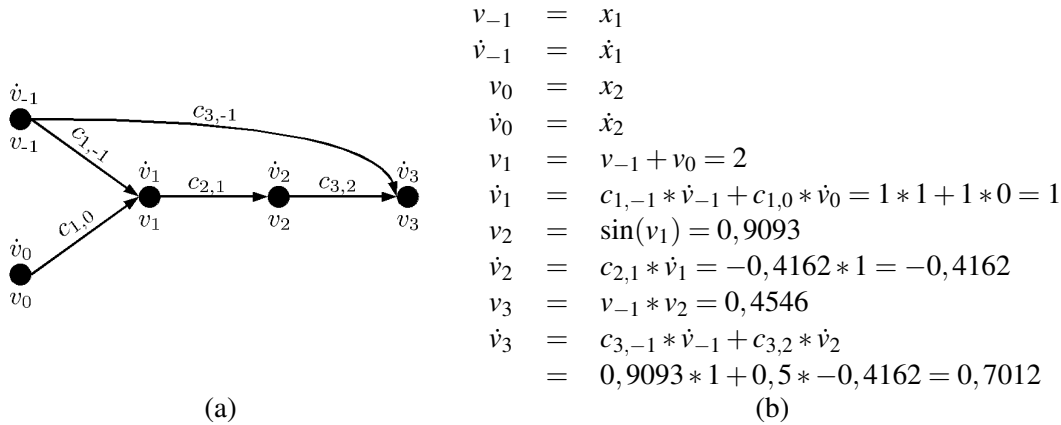
Die zu transformierende Funktion muss aus Zuweisungen mit jeweils nur einem Elementaroperator ϕ , wie z.B. $+$, $-$, $*$, $/$, $\sin()$, $\cos()$, bestehen, weil von diesem die Elementarableitung(en) bekannt sind. Falls eine Zuweisung diese Bedingung nicht erfüllt, kann sie mit Hilfe von zusätzlichen Zwischenvariablen in eine solche Form überführt werden. Die Zuweisungen $v_j = \phi_j(v_i)_{i < j}$ können unäre und binäre Elementaroperatoren sein, so dass es ein oder zwei Variablen v_i auf der rechten Seite gibt. Die Variable v_j bekommt also den Wert zugewiesen, der sich ergibt, wenn die Operation ϕ_j auf die eine oder zwei Variable(n) v_i der rechten Seite angewendet wird.

Eine Funktion, deren Zuweisungen jeweils nur aus einem Elementaroperator bestehen, kann als Berechnungsgraph $G = (V, E)$ dargestellt werden. In so einem Graphen gibt es für jede Variable einen Knoten in der Knotenmenge V . Die Kanten aus der Kantenmenge E geben an, welche Variablen direkt voneinander abhängen. Die Knoten v_{1-n}, \dots, v_0 entsprechen den Eingabevariablen x_1, \dots, x_n und die Knoten v_{q+1}, \dots, v_{q+m} den Ausgabevariablen y_1, \dots, y_m . In Abb. 2.1.1 können wir einen Berechnungsgraphen für die Funktion $f(x_1, x_2) = x_1 * \sin(x_1 + x_2)$ (zerlegte Form) betrachten.

Zu jeder Variable v_j , $j = 1, \dots, r = q + m$, muss nun ein Ableitungsobjekt \dot{v}_j berechnet werden. Dazu werden den Variablen v_{1-n}, \dots, v_0 die Werte x_1, \dots, x_n des Eingabevektors x und den Ableitungsobjekten $\dot{v}_{1-n}, \dots, \dot{v}_0$ die Werte $\dot{x}_1, \dots, \dot{x}_n$ des Seed-Vektors \dot{x} zugewiesen. Danach werden die Elementarableitungen $c_{j,i} = \frac{\delta \phi_j}{\delta v_i}(v_k)_{k < j}$ zu jeder Elementarzuweisung $\phi_j(v_i)_{i < j}$ be-

φ_j	$c_{j,i_1} = \frac{\delta \varphi_j}{\delta v_{i_1}}$	$c_{j,i_2} = \frac{\delta \varphi_j}{\delta v_{i_2}}$
+	1	1
*	v_{i_2}	v_{i_1}
$\sin()$	$\cos(v_{i_1})$	–

Abbildung 2.1.2: Elementarableitungen der Operationen +, * und sin()


 Abbildung 2.1.3: (a) Berechnungsgraph zur Funktion $f(x_1, x_2) = x_1 * \sin(x_1 + x_2)$ mit zusätzlichen Objekten für FM (b) Berechnung der Richtungsableitung $\dot{y} = 0,7012$ zur Richtung $\dot{x} = (1; 0)^T$ am Punkt $x = (0, 5; 1, 5)^T$

rechnet. Die Elementarableitungen, die für die Beispielfunktion aus Abb. 2.1.1 benötigt werden, sind exemplarisch in Abb. 2.1.2 aufgeführt. Die Ableitungsobjekte \dot{v}_j werden wie folgt ermittelt:

$$\dot{v}_j = \sum_{(i,j) \in E} c_{j,i} \cdot \dot{v}_i.$$

Im FM können die Werte der Variablen und der Ableitungsobjekte in einem Vorwärtsdurchlauf berechnet werden. Mittels der Kettenregel und durch Propagation eines Richtungsvektors \dot{x} kann eine Richtungsableitung \dot{y} bestimmt werden.

In Abb. 2.1.3 (a) sind an die Knoten des Berechnungsgraphen zusätzlich die Ableitungsobjekte und an die Kanten die Elementarableitungen angehängt worden. Die Pfeile an den Kanten geben an, dass die Ableitungsobjekte im Vorwärtsdurchlauf berechnet werden können. In (b) ist ein Beispiel für die Berechnung der Richtungsableitung \dot{y} zum Richtungsvektor $\dot{x} = (1; 0)^T$ am Punkt $x = (0, 5; 1, 5)^T$ angegeben. Das Ergebnis lautet $\dot{y} = 0,7012$. Für die zweite Richtung $\dot{x} = (0; 1)^T$ hat die Richtungsableitung \dot{y} den Wert $-0,2081$.

Im Rückwärtsmodus wird die Originalfunktion (2.2) zu folgender Funktion erweitert:

$$\bar{f} : \mathbb{R}^{n+m} \rightarrow \mathbb{R}^n : \bar{x} = \bar{f}(x, \bar{y}) = (f'(x))^T \cdot \bar{y}, \quad (2.5)$$

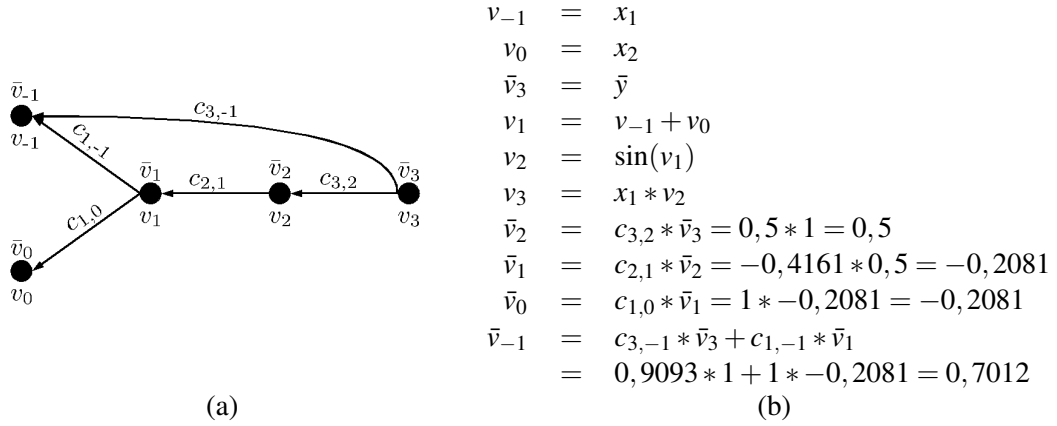


Abbildung 2.1.4: (a) Berechnungsgraph zu $f(x_1, x_2) = x_1 * \sin(x_1 + x_2)$ mit zusätzlichen Objekten für RM (b) Berechnung der Adjungierten $\bar{x} = (0,7012, -0,2081)$ zu der adjungierten Richtung $\bar{y} = 1$ am Punkt $x = (0,5; 1,5)^T$

wobei die adjungierte Richtung $\bar{y} \in \mathbb{R}^m$ einmal mit jedem der kanonischen Einheitsvektoren $e_i = (0, \dots, 1, \dots, 0)^T$, $i = 1, \dots, m$, geseedet wird, um die Jacobi-Matrix J zu bestimmen. In einem Durchlauf der Funktion kann nun eine Adjungierte $\bar{x} = (\frac{\delta y_i}{\delta x_1}, \dots, \frac{\delta y_i}{\delta x_n})$ (Zeile von J) berechnet werden.

Zur Bestimmung einer Adjungierten müssen folgende Schritte ausgeführt werden: Den Variablen v_{1-n}, \dots, v_0 werden die Werte x_1, \dots, x_n des Eingabevektors x zugewiesen. Dann werden im Vorwärtslauf die Werte der Knoten v_1, \dots, v_r und die Elementarableitungen $c_{j,i}$ ermittelt. Danach werden den Ableitungsobjekten $\bar{v}_{q+1}, \dots, \bar{v}_{r=q+m}$ die Werte $\bar{y}_1, \dots, \bar{y}_m$ des Seed-Vektors \bar{y} zugewiesen. Nun wird der Kontrollfluss umgekehrt und im Rückwärtslauf werden alle adjungierten Objekte von \bar{v}_j , $j = q, \dots, 1-n$, zu einer adjungierten Richtung \bar{y} am Punkt x berechnet. Die adjungierten Objekte \bar{v}_j werden wie folgt berechnet:

$$\bar{v}_j = \sum_{(j,k) \in E} c_{k,j} \cdot \bar{v}_k$$

Den Berechnungsgraphen zu der Beispielfunktion $f(x_1, x_2) = x_1 * \sin(x_1 + x_2)$ mit adjungierten Objekten und Elementarableitungen für den RM können in Abb. 2.1.4 (a) betrachtet werden. Die umgekehrten Richtungen der Kanten sollen kennzeichnen, dass der Graph zur Berechnung der Adjungierten rückwärts durchlaufen werden muss. In (b) wird die Adjungierte $\bar{x} = (0,7012, -0,2081)$ zur adjungierten Richtung $\bar{y} = 1$ am Punkt $x = (1,5; 0,5)^T$ ermittelt.

Somit kann festgestellt werden, dass für die Funktion $f(x_1, x_2) = x_1 * \sin(x_1 + x_2)$ im FM und im RM jeweils die Jacobi-Matrix

$$J|_x = \begin{pmatrix} -0,2081 & 0,7012 \end{pmatrix}$$

berechnet wird.

Ein Nachteil vom AD gegenüber den FD besteht darin, dass nicht der Originalprogrammcode zur Berechnung genutzt werden kann, sondern eine erweiterte Funktion dafür erstellt werden muss, die eine Richtungsableitung bzw. eine Adjungierte berechnet.

Der Aufwand hierbei hängt stark vom verwendeten Modus ab. Beim Vorwärtsmodus muss einmalig die beschriebene semantische Transformation vorgenommen werden, so dass zu jeder numerischen Operation die jeweilige Ableitung bestimmt wird. Nach dem Berechnen der modifizierten Funktion ist der Aufwand im FM nur gering höher (konstanter Faktor) als bei den FD.

Anhand der Dimensionen der Funktionen \hat{f} und \tilde{f} aus (2.4) und (2.5) können wir erkennen, dass das Verhältnis der Eingabe- zu den Ausgabevariablen entscheidend dafür ist, ob der FM oder der RM genutzt werden sollte. Beim FM werden n und beim RM werden m AD-Berechnungsdurchläufe benötigt, wenn die Richtungsvektoren mit den kanonischen Einheitsvektoren geseedet werden. Bei unserer Beispielfunktion werden im FM zwei Durchläufe und im RM nur ein Durchlauf benötigt.

Da der RM aufwendiger ist, weil viele Zwischenwerte gespeichert oder neu berechnet werden müssen, sollte die Anzahl der Ausgabevariablen gegenüber der Anzahl der Eingabevariablen signifikant höher sein. Dieses Kriterium kann für vollbesetzte Jacobi-Matrizen vernachlässigt werden, wenn sie quadratisch sind. Der FM wird deshalb bevorzugt.

Da die Jacobi-Matrix in unserem Fall jedoch dünnbesetzt ist, kann es vorkommen (auch im Zusammenhang mit den später im Kapitel vorgestellten *Techniken zum Seeden*), dass beim FM (RM) sehr viel weniger als n (m) Berechnungsdurchläufe benötigt werden. Dadurch kann es sinnvoll sein, nur den RM oder eine Kombination aus FM und RM zu verwenden. Hierauf wird in dem Abschnitt 2.3 (*Techniken zum Seeden*) näher eingegangen.

2.2 Vorkonditionierung

In [1, 21] wird die Technik der Vorkonditionierung (VK) vorgestellt. Sie wird angewendet, um die Konvergenz der iterativen Verfahren zum Lösen von linearen Gleichungssystemen zu erhöhen. Dadurch soll das Berechnen der Lösung eines Gleichungssystems beschleunigt werden. Allerdings muss der Vorkonditionierer M so gewählt werden, dass durch die benötigte Invertierung von M das Verfahren insgesamt nicht langsamer wird, obwohl die Konvergenz höher ist. Trotz der Vorkonditionierung soll die Lösung genau genug berechnet werden können.

Das Gleichungssystem (2.1) soll nun vorkonditioniert werden. Dafür gibt es mehrere Möglichkeiten:

$$M^{-1}Az = M^{-1}b, \quad (\text{links VK})$$

$$AM^{-1}y = b \quad \wedge \quad z = M^{-1}y \quad \text{und} \quad (\text{rechts VK})$$

$$M_L^{-1}AM_R^{-1}y = M_L^{-1}b \quad \wedge \quad z = M_R^{-1}y, \quad M = M_L \cdot M_R, \quad (\text{beidseitige VK})$$

wobei die Matrix $M \in \mathbb{R}^{n \times n}$ der Vorkonditionierer ist.

Somit ergibt sich nun folgender zusätzlicher Aufwand:

- Berechnung der Matrix M ,
- Lösen von $Mu = v$ (implizite VK) und/oder
- Berechnung des Matrix-Vektor-Produkts $z = Mv$ (explizite VK).

Bei der Wahl der Matrix M muss ein Kompromiss zwischen Konvergenz und Rechenaufwand gefunden werden. Es ist wichtig, dass das Gleichungssystem $Mu = v$ mit geringem Aufwand gelöst werden kann. Die beiden Extremfälle bei der Wahl von M sind folgende:

1. $M = E$ (Einheitsmatrix): Kein Berechnungsaufwand, aber auch keine Verbesserung der Konvergenz, da identisch zum Originalsystem.
2. $M = A$: Hoher Berechnungsaufwand, allerdings Konvergenz nach einem Schritt.

Bei bestimmten Gleichungssystemen ist zum Beispiel die Matrix M , bei der nur die Diagonalelemente von A vorhanden sind, eine gute Wahl. Dieser Fall ist ein gutes Beispiel für die Notwendigkeit der partiellen Berechnung von Jacobi-Matrizen (vgl. Abschnitt 2.5).

2.3 Techniken zum Seeden

Zur Bestimmung einer dünnbesetzten Jacobi-Matrix J sind bis jetzt n AD-Berechnungsdurchläufe im FM für die n Richtungsableitungen der Jacobi-Matrix benötigt worden, da die Richtungsvektoren mit den kanonischen Einheitsvektoren geseedet wurden. Für die Adjungierten werden entsprechend m AD-Berechnungsdurchläufe im RM benötigt.

Es soll nun gezeigt werden, dass es oftmals möglich ist, eine Linearkombination von zwei oder mehrere Richtungsableitungen bzw. Adjungierte gleichzeitig in einem AD-Durchlauf zu berechnen. Somit wäre der Aufwand zur Berechnung von zwei oder mehreren Richtungsableitungen (Adjungierten) nur gering höher als die Berechnung einer Richtungsableitung (Adjungierten).

Der Einfachheit halber beziehen wir uns im Folgenden nur auf das Zusammenfassen der Spalten einer Jacobi-Matrix J , da dieses dem Zusammenfassen der Zeilen von J^T entspricht. Eine Ausnahme ist der Fall, in dem die Jacobi-Matrix durch eine Kombination aus Richtungsableitungen und Adjungierten ermittelt werden soll.

Zur gleichzeitigen Berechnung mehrerer Richtungsableitungen wird eine Seed-Matrix bestimmt, bei der die Seed-Vektoren – entsprechend der Richtungsableitungen – so weit wie möglich zusammengefasst sind. Um die komprimierte Jacobi-Matrix \tilde{J} zu erhalten, muss zu jedem

Vektor der Seed-Matrix S (Seed-Vektor) eine Richtungsableitung berechnet werden. Die Jacobi-Matrix kann hieraus durch das Lösen bestimmter Gleichungssysteme ermittelt werden. Durch die gleichzeitige Berechnung mehrerer Ableitungen wird eine komprimierte Jacobi-Matrix \tilde{J} mit p Durchläufen, $p < n$, bestimmt.

Zur Berechnung einer Seed-Matrix muss bekannt sein, welche Nichtnullelemente eine Richtungsableitung haben wird. Die Matrix $B \in \mathbb{R}^{n \times n}$ sei die Dünnesetztheitsstruktur einer Jacobi-Matrix J . Diese Matrix gibt die Abhängigkeit der Ausgabe- von den Eingabevariablen an. Dieses Muster kann beispielsweise durch *Operator-Overloading* bestimmt werden, indem jeder Variablen ein boolescher Vektor der Länge n zugewiesen wird. Während einer Ausführung der Funktion wird in jedem dieser Vektoren die Abhängigkeit der entsprechenden Variable von den Eingabevariablen gespeichert.

Folgende Schritte müssen also zur Berechnung einer komprimierten Jacobi-Matrix \tilde{J} und zur Bestimmung der Jacobi-Matrix J durchgeführt werden:

1. Bestimmung einer Seed-Matrix S
2. Berechnung der komprimierten Jacobi-Matrix \tilde{J}
3. Bestimmung der Jacobi-Matrix J aus der komprimierten Jacobi-Matrix \tilde{J}

Es gibt mehrere Techniken, Seed-Matrizen zu berechnen. In [17] wird das Prinzip der direkten Verfahren, der Substitutions- und der Eliminationsverfahren vorgestellt. In der vorliegenden Arbeit wird nur eine direkte Methode vorgestellt. Auf die übrigen Verfahren wird nur kurz hingewiesen.

Bei den **direkten Verfahren** (direct methods) werden die Spalten nur soweit zusammengefasst, dass jedes Element der Jacobi-Matrix J *direkt* aus der komprimierten Jacobi-Matrix \tilde{J} bestimmt werden kann. D.h. es muss kein Element von J aus mehreren Elementen von \tilde{J} ermittelt werden.

Zwei Richtungsableitungen dürfen gleichzeitig berechnet werden, wenn in keiner Zeile beide Richtungsableitungen ein Nichtnullelement haben. D.h. keine zwei Richtungsableitungen sind von mehreren Richtungen abhängig. Damit kommen wir direkt zur Definition:

Definition 2.3.1. Gegeben sei eine beliebige Matrix. Zwei Spalten dieser Matrix sind *strukturell orthogonal*, wenn in jeder gleichen Zeile höchstens ein Nichtnullelement existiert. (Vgl. [13])

Demnach dürfen Richtungsableitungen, deren Besetzungsmuster paarweise strukturell orthogonal sind, in einem AD-Durchlauf berechnet werden. Wir betrachten Abb. 2.3.1, um diese Eigenschaft zu erklären. Die Spalten 1 und 2 von J sind strukturell orthogonal, da sie in keiner Zeile beide ein Element besitzen. Für die Spalten 2 und 3 gilt das nicht, da sie beide in der dritten Zeile ein Element haben.

Bei dem Partitionieren einer Matrix geht es darum, Spalten oder Zeilen zu Gruppen zusammenzufassen, wobei in einer Gruppe niemals gleichzeitig Spalten und Zeilen enthalten sein

$$\begin{array}{ccc}
 \begin{pmatrix} \tilde{j}_{1,1} & & \\ \tilde{j}_{2,2} & \tilde{j}_{2,5} & \\ \tilde{j}_{3,2} & \tilde{j}_{3,3} & \tilde{j}_{3,4} \\ & \tilde{j}_{4,3} & \tilde{j}_{4,4} \\ & & \tilde{j}_{5,5} \end{pmatrix} & = & \begin{pmatrix} j_{1,1} & & & & \\ & j_{2,2} & & & j_{2,5} \\ & j_{3,2} & j_{3,3} & j_{3,4} & \\ & & j_{4,3} & j_{4,4} & \\ & & & & j_{5,5} \end{pmatrix} * \begin{pmatrix} 1 & & \\ & 1 & \\ & & 1 \end{pmatrix} \\
 \tilde{J} & & J \qquad \qquad S
 \end{array}$$

Abbildung 2.3.1: Die Seed-Matrix S gibt an, welche Richtungsableitungen gleichzeitig berechnet werden können. Wenn die Spalten von S als Richtungsvektoren genutzt werden, erhalten wir die entsprechenden Richtungsableitungen als komprimierte Jacobi-Matrix \tilde{J} . Von J sind nur die Positionen der Nichtnullelemente bekannt.

dürfen. Wenn nur die Spalten oder Zeilen partitioniert werden, wird die Partitionierung unidirektional genannt. Wenn sowohl Spalten als auch Zeilen zu Gruppen zusammengefasst werden, dann wird von bidirektionaler Partitionierung gesprochen. Alle Spalten bzw. Zeilen, die in derselben Gruppe einer Partition sind, können gleichzeitig in einem AD-Durchlauf des FM bzw. RM berechnet werden.

Eine konsistente Partition (alle Spalten oder Zeilen einer Gruppe sind strukturell orthogonal) können wir anhand der Matrix B in Abb. 2.3.2 erkennen. Diese Matrix B ist das Dünnbesetzmuster der Jacobi-Matrix J aus Abb. 2.3.1. Die Partition enthält drei Spaltengruppen $\{1, 2\}$ (grün), $\{3, 5\}$ (blau) und $\{4\}$ (rot), die jeweils unterschiedlich gefärbt sind.

Damit kommen wir direkt zur Problemdefinition:

Problem 2.3.2. Gegeben sei die Struktur einer Jacobi-Matrix J als Matrix B . Gesucht wird eine strukturell orthogonale Partition der Spalten mit der geringsten Anzahl von Gruppen. (Vgl. [13, Problem 3.3])

Nun wird ein direktes Verfahren zur Berechnung einer Seed-Matrix vorgestellt und gezeigt, wie es möglich ist, die Elemente der Jacobi-Matrix aus der komprimierten Jacobi-Matrix zu bestimmen.

Zur Bestimmung einer konsistenten Partition wird die so genannte CPR-Technik [9] von Curtis, Powell und Reid verwendet. Pro Schritt wird eine Spalte betrachtet. Diese Spalte wird der ersten Gruppe hinzugefügt, in der sie zu allen enthaltenen Spalten strukturell orthogonal ist. Gibt es keine solche Gruppe, wird für diese Spalte eine neue Gruppe angelegt.

Danach wird die Einheitsmatrix E_n genauso partitioniert, um die Seed-Matrix zu bestimmen. Als Beispiel betrachten wir die partitionierte Matrix B , die partitionierte Einheitsmatrix E_n und die resultierende Seed-Matrix S in Abb. 2.3.2.

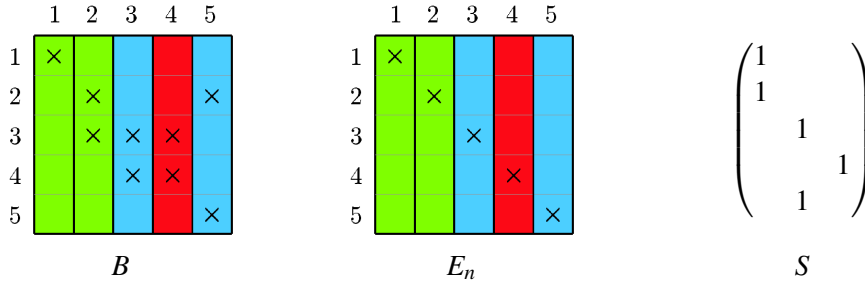


Abbildung 2.3.2: B : Partition von Matrix B (Dünnbesetztheitsstruktur einer Jacobi-Matrix J) mit drei Spaltengruppen (grün, blau und rot); E_n : Einheitsmatrix, die wie B partitioniert ist; S : resultierende Seed-Matrix

$$\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} * \begin{pmatrix} j_{2,2} \\ j_{2,5} \end{pmatrix} = \begin{pmatrix} \tilde{j}_{2,2} \\ \tilde{j}_{2,5} \end{pmatrix} \quad (2.6)$$

Abbildung 2.3.3: Zur Bestimmung der Elemente der zweiten Zeile der Matrix J aus Abb. 2.3.1, muss das Gleichungssystem $\hat{S}_2^T \cdot \mathbf{j}_2^T = \tilde{\mathbf{j}}_2^T$ gelöst werden.

Nun werden so viele AD-Berechnungsdurchläufe durchgeführt wie die Seed-Matrix Spalten hat, da eine Spalte als ein Seed-Vektor genutzt wird. Zum Schluss muss die Jacobi-Matrix wie folgt bestimmt werden:

In [17] wird beschrieben, wie jede Spalte der Jacobi-Matrix einzeln aus der komprimierten Jacobi-Matrix bestimmt werden muss. Dazu wird für jede Zeile i eine reduzierte Seed-Matrix \hat{S}_i bestimmt, indem \hat{S}_i für jedes Nichtnullelement $j_{i,k}$ von J die k -te Zeile der Matrix S erhält. Danach wird das Gleichungssystem $\hat{S}_i^T \cdot \mathbf{j}_i^T = \tilde{\mathbf{j}}_i^T$ gelöst, wobei die Zeile \mathbf{j}_i von J und die Zeile $\tilde{\mathbf{j}}_i$ von \tilde{J} nur die Nichtnullelemente enthalten. Die Abbildung 2.3.3 zeigt ein Beispiel für die Bestimmung der zweiten Zeile der Jacobi-Matrix J aus Abb. 2.3.1. Hier kann erkannt werden, dass das Gleichungssystem nicht explizit gelöst werden muss, sondern dass eine direkte Entnahme der Elemente möglich ist.

Des Weiteren ist es möglich, die Elemente einer Jacobi-Matrix J durch eine Kombination aus Spalten- und Zeilenberechnung zu erhalten. Es gibt Fälle, in denen hierbei weniger Gruppen als bei der unidirektionalen Partitionierung bzgl. der Spalten oder Zeilen benötigt werden. In Abb. 2.3.4 ist eine Spaltenpartition der Matrix B_{uni} (Dünnbesetztheitsmuster von J) und eine Kombination aus Spalten- und Zeilenpartition der Matrix B_{bi} zu sehen. Bei der unidirektionalen Partitionierung werden fünf Gruppen ($\{\text{Spalte } 1\}$ (grün), $\{2\}$ (blau), $\{3\}$ (violett), $\{4\}$ (rot) und $\{5\}$ (gelb)) und bei der bidirektionalen Partitionierung nur 4 Gruppen benötigt, die sich aus einer Zeilengruppe ($\{1\}$ (violett)) und drei Spaltengruppen ($\{1\}$ (grün), $\{2, 3\}$ (blau), und $\{4, 5\}$ (rot)) zusammensetzen.

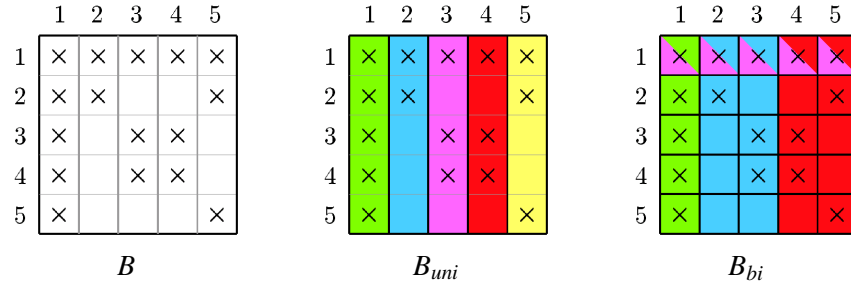


Abbildung 2.3.4: B : Dünnesetztheitsmuster einer Matrix J ; B_{uni} : Undirektionale Partition von B mit fünf Spaltengruppen (minimal); B_{bi} : Bidirektionale Partition von B mit vier Gruppen (minimal)

Das Problem des bidirektionalen Partitionierens zur direkten Bestimmung der Elemente der Jacobi-Matrix wird wie folgt beschrieben:

Problem 2.3.3. Gegeben sei die Struktur einer Jacobi-Matrix J als Matrix B . Gesucht werden eine Matrix $S_c \in \{0, 1\}^{n \times p_1}$ und eine Matrix $S_r \in \{0, 1\}^{p_2 \times n}$, so dass die Produkte $J \cdot S_c$ und $S_r \cdot J$ die direkte Bestimmung von J ermöglichen und der Wert von $p = p_1 + p_2$ minimal ist. (Vgl. [13, Problem 5.1])

Bei den **Substitutionsverfahren** (substitution methods) [16] können oftmals noch mehr Spalten als bei den direkten Verfahren zusammengefasst werden. Somit können weitere AD-Berechnungsdurchläufe gespart werden. Bei diesen Verfahren müssen die Spalten der Matrix, die zusammengefasst werden, nicht strukturell orthogonal sein. Als einzige Bedingung muss gelten, dass die Elemente durch das Lösen von linearen Gleichungssystemen, bei denen die reduzierte Seed-Matrix jeweils triangular ist, bestimmt werden können. Hierbei können beim Lösen der Gleichungssysteme numerische Ungenauigkeiten auftreten.

Eine noch stärkere Zusammenfassung ist bei der letzten Kategorie dieser Verfahren, den **eliminierenden Verfahren** (elimination methods) [17], möglich. Hierbei wird die Bedingung aufgehoben, dass die reduzierten Seed-Matrizen beim Lösen der Gleichungssysteme triangular sein müssen. Sie müssen nur noch regulär sein.

Nachdem die Seed-Matrix S in einem Substitutions- bzw. Eliminationsverfahren bestimmt und die komprimierte Jacobi-Matrix \tilde{J} berechnet worden ist, können die Elemente der Jacobi-Matrix J aus der Matrix \tilde{J} – wie beim direkten Verfahren beschrieben – ermittelt werden. Allerdings muss nun tatsächlich ein Gleichungssystem gelöst werden. Eine direkte Entnahme der Elemente aus der komprimierten Jacobi-Matrix \tilde{J} ist nicht mehr möglich. (Vgl. dazu [17])

Es gibt noch weitere Techniken zur Bestimmung von Seed-Matrizen, wie z.B. das Verfahren *column segments* oder eine Kombination aus den verschiedenen Verfahren. Auf diese wird aber nicht weiter eingegangen.

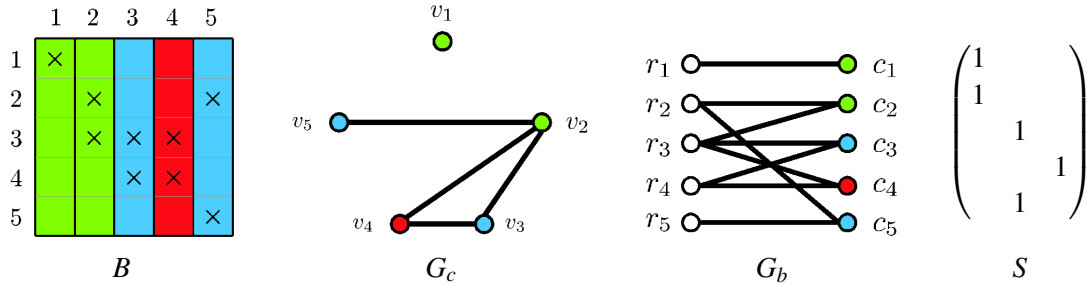


Abbildung 2.4.1: Matrix B ist das Dünnsbesetztheitsmuster einer Jacobi-Matrix; G_c ist der CIG von B ; G_b ist der bipartite Graph von B ; die Partitionierung von B und die Färbungen von G_c und G_b führen alle zur Seed-Matrix S

2.4 Einführung in die Graphfärbung

Im vorherigen Abschnitt haben wir verschiedene Techniken zum Seeden kennengelernt. Nun soll gezeigt werden, wie es durch das Färben von Graphen möglich ist, Seed-Matrizen zu bestimmen.

Auf das Färben von Graphen gehen wir später in diesem Abschnitt ein. Zuerst wird gezeigt, wie eine Matrix bzw. bestimmte ihrer Eigenschaften als Graph dargestellt werden können und wie die CPR-Technik als Graphfärbung betrachtet werden kann.

Es gibt verschiedene Möglichkeiten, eine Matrix bzw. bestimmte ihrer Eigenschaften durch einen Graphen darzustellen. Zuerst wird der „column intersection graph“ (CIG) und danach der bipartite Graph einer Matrix betrachtet. Abbildung 2.4.1 enthält ein Beispiel dafür.

Definition 2.4.1. Der „column intersection graph“ (CIG) $G_c(B) = (V, E)$ einer $m \times n$ Matrix B sei wie folgt definiert:

- $V = \{v_i : 1 \leq i \leq n\}$,
- $E = \{(v_i, v_j) : \exists k (b_{k,i} \neq 0 \wedge b_{k,j} \neq 0)\}$.

Für jede Spalte der Matrix B existiert ein Knoten in V von G_c . Zwei Knoten, deren entsprechende Spalten nicht strukturell orthogonal sind, werden durch eine Kante verbunden. (Vgl. [13, S. 645])

Eine weitere Darstellungsmöglichkeit ist der bipartite Graph. Hierbei wird im Gegensatz zum CIG nicht nur eine Eigenschaft modelliert, sondern der bipartite Graph ist eine eindeutige Darstellung der Dünnsbesetztheitsstruktur einer Matrix.

Definition 2.4.2. Der bipartite Graph $G_b(B) = (V_r, V_c, E)$ einer $m \times n$ Matrix B sei wie folgt definiert:

- $V_r = \{r_i : 1 \leq i \leq m\}$,

- $V_c = \{c_j : 1 \leq j \leq n\}$,
- $E = \{(r_i, c_j) : b_{i,j} \neq 0\}$.

Für jede Spalte und Zeile der Matrix B existiert ein entsprechender Knoten in V_r bzw. V_c . Für jedes Element $b_{i,j}$ aus B werden die Knoten r_i und c_j durch eine Kante miteinander verbunden. (Vgl. [13, S. 644])

Dadurch, dass hier nur quadratische Matrizen betrachtet werden, sind die beiden Knotenmengen der bipartiten Graphen jeweils gleich groß.

In [6] wird gezeigt, dass das Problem 2.3.2 auch als Graphfärbungsproblem betrachtet werden kann, bei dem eine sogenannte *Distanz-1 Färbung* des CIG mit minimaler Farbanzahl gesucht wird. Eine Färbung mit minimaler Farbanzahl wird im Folgenden auch minimale Färbung genannt. So eine *Distanz-1 Färbung* eines CIG entspricht der Partitionierung der entsprechenden Matrix mit der CPR-Technik. Die CPR-Technik kann wie folgt auf den CIG übertragen werden: Adjazente Knoten erhalten unterschiedliche Farben, wenn die entsprechenden Spalten nicht strukturell orthogonal sind. Somit können zwei Spalten der Jacobi-Matrix gleichzeitig berechnet werden, wenn den entsprechenden Knoten dieselbe Farbe zugewiesen worden ist.

Zuerst wird die Distanz- k Färbung definiert, um anschließend das p -Färbbarkeitsproblem zu beschreiben. Dieses Problem ist eine Generalisierung des im vorherigen Absatz beschriebenen Färbbarkeitsproblems, weil nicht nur adjazente Knoten, die über einen Pfad der Länge zwei miteinander verbunden sind, betrachtet werden, sondern auch Knoten, die über einen Pfad der Länge k miteinander verbunden sind. Danach wird das Problem der *Distanz-1 Färbung des CIG* angegeben.

Definition 2.4.3. (Distanz- k Färbung) Für eine Distanz- k Färbung werden die Knoten des Graphen $G = (V, E)$ so gefärbt, dass allen Knoten, die durch einen Pfad der Länge l , $l \leq k$, miteinander verbunden sind, unterschiedliche Farben zugewiesen werden. (Vgl. [13, Abschnitt 3.2])

Problem 2.4.4. (p -Färbbarkeitsproblem) Gegeben sei ein Graph $G = (V, E)$. Existiert eine Distanz- k Färbung von G mit $p \geq 3$ Farben?

Die Beschränkung $p \geq 3$ ist notwendig, da es für den Fall $p = 2$ einen polynomiellen Algorithmus gibt und der Fall $p = 1$ trivial ist, da der Graph hier keine Kanten haben darf. Der polynomielle Algorithmus funktioniert wie folgt: In jedem Schritt wird ein Knoten $v \in V$ besucht. Dabei werden alle bereits gefärbten Nachbarn betrachtet. Wenn alle diese Nachbarknoten identisch gefärbt sind, dann erhält v die andere Farbe. Dann wird der nächste Knoten v betrachtet, bis alle Knoten einmal besucht worden sind. Wenn allerdings zwei dieser Nachbarknoten unterschiedlich gefärbt sind, wird abgebrochen, da es keine Färbung mit zwei Farben gibt. Eine obere Abschätzung der Laufzeit ist somit $O(|E|)$, da jede Kante nur einmal betrachtet wird.

Das Problem der Distanz-1 Färbung für einen CIG (p -Färbbarkeitsproblem auf dem CIG) kann wie folgt definiert werden:

Problem 2.4.5. Gegeben sei der CIG einer $m \times n$ Matrix B . Gesucht ist eine *Distanz-1 Färbung* von $G_c(B)$ mit den wenigsten Farben $p \geq 3$. (Vgl. [6])

Im Graphen G_c in Abb. 2.4.1 müssen z.B. die Knoten $v_2, v_3 \in V$ unterschiedlich gefärbt werden, da die beiden Knoten adjazent sind.

Die Seed-Matrix S kann wie folgt aus einer solchen Färbung bestimmt werden: Für jede Farbe erhält S eine Spalte. Die Matrix S hat dieselbe Anzahl an Zeilen wie die Matrix B bzw. J . Wenn ein Knoten $v_i \in V$ mit der Farbe k gefärbt ist, wird dem Element $s_{i,k}$ der Matrix S der Wert 1 zugewiesen. Als Beispiel betrachten wir Abb. 2.4.1. Die Knoten $v_1, v_2 \in V$ im CIG G_c sind beide mit der Farbe grün gefärbt. Darum hat die erste Spalte von S eine 1 in der ersten und zweiten Komponente.

Die zweite Möglichkeit, um eine unidirektionale Partition durch eine Graphfärbung zu berechnen, ist die Einfärbung des bipartiten Graphen. Hierbei wird nur eine der beiden Knotenmengen gefärbt. Den Knoten dieser Menge müssen die Farben so zugewiesen werden, dass alle Distanz-2 Nachbarn unterschiedliche Farben erhalten, wobei zwei *Distanz- k Nachbarn* über einen Pfad der Länge k miteinander verbunden sind. Die gesuchte Färbung und das entsprechende Problem sind wie folgt definiert:

Definition 2.4.6. Eine Färbung eines bipartiten Graphen $G_b = (V_r, V_c, E)$, bei der nur die Knoten aus V_r oder nur aus V_c gefärbt werden, wird *einseitige Färbung von G_b auf V_r bzw. V_c* genannt. Wenn die Distanz-2 Nachbarn einer der Mengen unterschiedlich gefärbt werden, dann liegt eine *einseitige Distanz-2 Färbung von G_b auf V_r bzw. V_c* vor. (Vgl. [13, Abschnitt 2.2])

Problem 2.4.7. Sei $G_b = (V_r, V_c, E)$ ein bipartiter Graph mit $|V_r| = |V_c|$. Gesucht ist eine *einseitige Distanz-2 Färbung von G_b auf V_c* mit den wenigsten Farben $p \geq 3$. (Vgl. [13, Problem 3.6])

In [13, Abschnitt 3.3] wird gezeigt, dass die Distanz-1 Färbung von $G_c(B)$ der *einseitigen Distanz-2 Färbung von $G_b(B)$ auf V_c* entspricht. Anhand von Abb. 2.4.1 können wir exemplarisch feststellen, dass die Knoten $c_2, c_3 \in V_c$, die den Knoten $v_2, v_3 \in V$ entsprechen, auch unterschiedlich gefärbt sind, da c_2 und c_3 Distanz-2 Nachbarn sind.

Nun wird das bidirektionale Partitionieren der Dünnbesetztheitsstruktur einer Jacobi-Matrix betrachtet. Ein entsprechendes Graphfärbungsproblem muss auf dem bipartiten Graphen definiert werden, da der CIG dafür nicht ausdrucksstark genug ist.

Beim bidirektionalen Partitionieren werden zwei Seed-Matrizen S_r und S_c bestimmt. Die Matrix S_c gibt an, welche Richtungsableitungen gemeinsam berechnet werden können. Die Matrix S_r gibt dieses für die Adjungierten an. Dazu müssen nun beide Knotenmengen des bipartiten Graphen gefärbt werden. Hierbei wird versucht, insgesamt eine möglichst geringe Farbanzahl zu erhalten, damit möglichst wenige AD-Berechnungsdurchläufe benötigt werden. Die Knoten, die einer Spalte (Richtungsableitung) bzw. einer Zeile (Adjungierte) entsprechen und nicht für die Berechnung der Elemente relevant sind, bekommen die Farbe 0 zugewiesen. Die Elemente dieser Spalten und/oder Zeilen werden somit nicht berechnet.

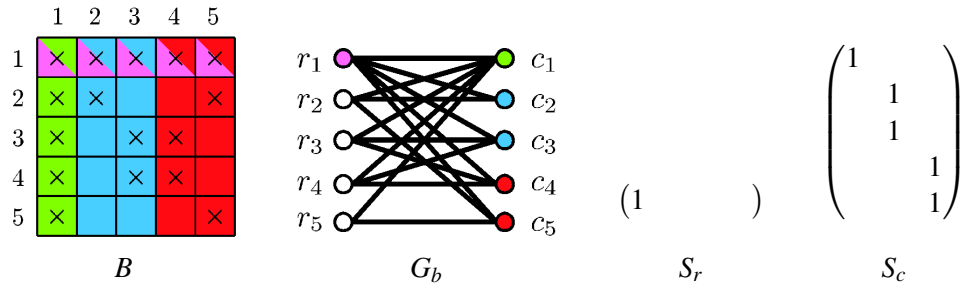


Abbildung 2.4.2: Matrix B ist Dünnsbesetztheitsmuster einer Jacobi-Matrix J ; G_b ist der bipartite Graph zu B ; die Partitionierung von B und die Färbung von G_b führen zu den Seed-Matrizen S_r und S_c

In Abb. 2.4.2 sehen wir die partitionierte Matrix B (Matrix B_{bi} aus Abb. 2.3.4). Daneben ist der entsprechende bipartite Graph G_b mit einer Färbung dargestellt, die der konsistenten Partition von B entspricht, dargestellt. Zusätzlich sind noch die resultierenden Seed-Matrizen S_r und S_c angegeben.

2.5 Partielle Berechnung von Jacobi-Matrizen

Es gibt Fälle, in denen eine Jacobi-Matrix J nicht vollständig, sondern nur partiell berechnet werden muss. D.h., dass es ausreicht, wenn nur ein Teil der Elemente direkt aus der komprimierten Jacobi-Matrix \tilde{J} bestimmt werden kann. Diese partielle Berechnung ist z.B. hilfreich, wenn für einen Vorkonditionierer nur bestimmte Elemente benötigt werden. Es gibt z.B. Fälle, in denen nur die Elemente auf der Hauptdiagonalen einer Jacobi-Matrix als Vorkonditionierer benutzt werden (vgl. Abschnitt 2.2).

Die Einträge, die direkt bestimmt werden müssen, werden *benötigte Elemente* genannt. Die nicht relevanten Einträge heißen *nicht benötigte Elemente*. Bei der vollständigen Berechnung gibt es nur benötigte Elemente.

Gegeben sei die Dünnsbesetztheitsstruktur einer Jacobi-Matrix J als Matrix B . Die Menge R enthält die benötigten Elemente. Die Abb. 2.5.1 enthält ein Beispiel dafür: Die Menge der benötigten Elemente R der Matrix B_2 besteht aus folgenden Elementen: $R = \{b_{1,1}, b_{2,2}, b_{2,5}, b_{3,4}, b_{4,3}, b_{4,4}, b_{5,5}\}$. Die Kanten im bipartiten Graph $G_b(B_2)$, die den benötigten Elementen entsprechen, sind in der Menge $E_R = \{(r_1, c_1), (r_2, c_2), (r_2, c_5), (r_3, c_4), (r_4, c_3), (r_4, c_4), (r_5, c_5)\}$ enthalten.

Bei der partiellen Berechnung von Jacobi-Matrizen wird ein Knoten mit der Farbe 0 gefärbt, wenn dieser Knoten einer Spalte bzw. Zeile entspricht, die nicht berechnet werden muss, weil sie kein benötigtes Element enthält. Bei der partiellen Berechnung müssen nun die Spalten nicht mehr strukturell orthogonal sein, um in einer Gruppe zusammengefasst werden zu dürfen.

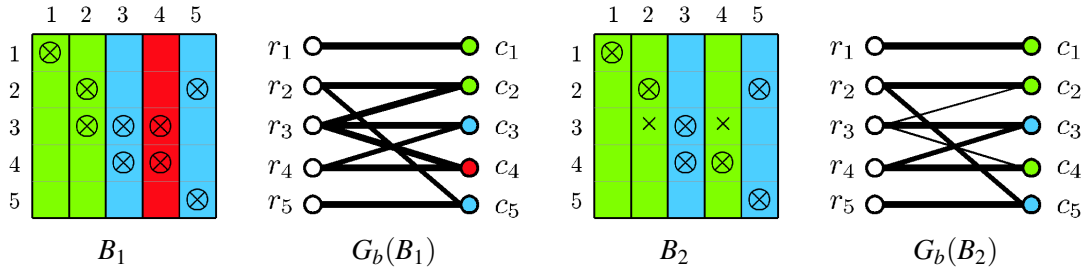


Abbildung 2.5.1: Matrizen B_1 und B_2 sind Dünnbesetzmuster von J mit unterschiedlichen benötigten Elementen \otimes und nicht benötigten Elementen \times ; dazu die entsprechenden bipartiten Graphen mit dicken Kanten für \otimes - und dünnen Kanten für \times -Elemente; für die Partitionierung von B_1 (G_b) werden drei Farben und für die von B_2 (G_b) zwei Farben benötigt

Zwei Spalten dürfen nicht gleichzeitig berechnet werden, wenn sie in einer Zeile zwei benötigte Elemente oder ein benötigtes und ein nicht benötigtes Element enthalten.

Als Beispiel dafür, dass weniger AD-Berechnungsdurchläufe bei der partiellen Berechnung möglich sind, betrachten wir die Abbildung 2.5.1. Für die Matrix B_1 (Dünnbesetzmuster einer Jacobi-Matrix J) gibt es folgende minimale Partition mit den Spaltengruppen $\{1, 2\}$ (grün), $\{3, 5\}$ (blau) und $\{4\}$ (rot). Zur partiellen Berechnung der Matrix B_2 (Dünnbesetzmuster von J) werden aber nur zwei Spaltengruppen ($\{1, 2, 4\}$ (grün), $\{3, 5\}$ (blau)) benötigt, da die Elemente $b_{3,2}$ und $b_{3,4}$ sich gegenseitig beeinflussen dürfen. Also werden für die partielle Berechnung der Jacobi-Matrix J nur zwei, für die vollständige Berechnung allerdings drei AD-Berechnungsdurchläufe benötigt.

Die Färbungen des bipartiten Graphen sind entsprechend anzupassen. Nun müssen nicht mehr alle Knoten, die über einen Pfad der Länge zwei miteinander verbunden sind, unterschiedlich gefärbt werden, sondern nur diejenigen, die durch einen Pfad der Länge zwei verbunden sind, von dem mindestens eine Kante in der Menge E_R enthalten ist. Genau wie die Spalte 4 der Matrizen B_1 und B_2 verhält es sich mit dem Knoten $c_4 \in V_c$ in den Graphen $G_b(B_1)$ und $G_b(B_2)$. Die Knoten $v_2, v_4 \in V_c$ dürfen in $G_b(B_2)$ gleich gefärbt werden, da der verbindende Pfad keine Kante aus E_R enthält. Im bipartiten Graphen $G_b(B_1)$ dürfen die entsprechenden Knoten jedoch nicht gleich gefärbt werden.

3 Färbungsprobleme

In diesem Kapitel werden Graphfärbungen von bipartiten Graphen und die entsprechenden Minimierungs- respektive Entscheidungsprobleme vorgestellt. Außerdem werden NP-Vollständigkeitsbeweise für bestimmte Entscheidungsprobleme angegeben. Abschließend wird ein Spezialfall der beschränkten Graphfärbung betrachtet.

Die Probleme der vollständigen und partiellen Berechnung einer Jacobi-Matrix J werden – wie im vorherigen Kapitel beschrieben – nicht als Partitionierungsprobleme, sondern als Graphfärbungsprobleme auf dem entsprechenden bipartiten Graph $G_b(J) = (V_r, V_c, E)$ betrachtet. Ein zu färbender Graph stellt die Dünnbesetztheitsstruktur einer Jacobi-Matrix dar. Alle Spalten bzw. Zeilen, deren entsprechende Knoten in dem Graphen mit derselben Farbe gefärbt werden, gehören zu derselben Gruppe der Partition. Aus einer solchen Partition erhält man entweder eine Seed-Matrix (unidirektionale Partitionierung) oder zwei Seed-Matrizen (bidirektionale Partitionierung).

Zuerst werden diejenigen Graphfärbungsprobleme betrachtet, deren Färbung zu einer Seed-Matrix S bzw. zwei Seed-Matrizen S_r und S_c führt, so dass die Jacobi-Matrix J direkt und vollständig aus der komprimierten Jacobi-Matrix $\tilde{J} = J \cdot S$ bzw. $\tilde{J} = S_r \cdot J \cdot S_c$ bestimmt werden kann. Anschließend werden dieselben Probleme im Kontext der partiellen Berechnung von Jacobi-Matrizen betrachtet. Die Färbungsprobleme hängen nach [12] von den folgenden Faktoren ab:

- Einseitige oder zweiseitige Färbung (uni- oder bidirektionale Partitionierung)
- Bestimmung der Elemente von J aus \tilde{J} direkt oder durch Substitution
- Vollständige oder beschränkte Färbung (vollständige oder partielle Berechnung von J)

Die Bedeutung dieser Eigenschaften ist in Kapitel 2 erläutert worden. In der vorliegenden Arbeit sind nur die direkten Verfahren von Interesse, so dass wegen der Orthogonalität der Faktoren die folgenden vier Probleme betrachtet werden: das EDV- (einseitig, direkt, vollständig), das EDB-, das ZDV- und das ZDB-Problem. Es wird gezeigt, dass alle diese Probleme NP-vollständig sind.

Die Komplexitätstheorie (allgemeine Theorie, NP-Vollständigkeit von Entscheidungsproblemen und NP-Schwierigkeit von Optimierungsproblemen) wird im Folgenden als bekannt vorausgesetzt. Diese wird z.B. in [11] behandelt.

Das Ziel der Minimierungsprobleme ist es, einen Graphen mit der minimalen Farbanzahl zu färben, damit so wenige AD-Berechnungsdurchläufe wie möglich für die Bestimmung einer

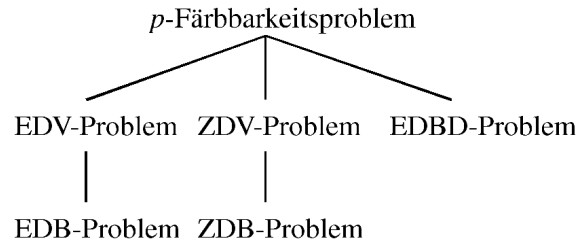


Abbildung 3.0.1: Übersicht der zu betrachtenden Probleme und der Reduktionsbaum, passend zu den folgenden NP-Vollständigkeitsbeweisen (die vertikalen Kanten bedeuten, dass das obere auf das untere Problem reduziert wird)

Jacobi-Matrix benötigt werden. Es wird gezeigt, dass die vier zu betrachtenden Graphfärbungsprobleme NP-schwierig sind.

Zusätzlich werden das EDBD-Problem und ZDBD-Problem betrachtet – spezielle Probleme, bei denen nur die Hauptdiagonalelemente einer Jacobi-Matrix benötigte Elemente sind. Es wird gezeigt, dass die minimalen Farbanzahlen beim EDBD- und beim ZDBD-Problem identisch sind, und dass das EDBD-Problem NP-vollständig ist.

In Abbildung 3.0.1 ist der Reduktionsbaum aller Reduktionen, die in diesem Kapitel durchgeführt werden, zu sehen.

In [7, 15] werden die Graphfärbungsprobleme betrachtet, deren Färbungen zu einer einzelnen Seed-Matrix (unidirektionale Partitionierung) bzw. zu zwei Seed-Matrizen (bidirektionale Partitionierung) führen, so dass die den bipartiten Graphen entsprechenden Jacobi-Matrizen direkt und vollständig bestimmt werden können. Die beschränkten Graphfärbungsprobleme, die der partiellen Bestimmung einer Jacobi-Matrix J entsprechen, sind zuerst in [12] und später darauf aufbauend in [13] definiert worden.

Wie in Kapitel 2 erklärt, sind in dem Kontext der vorliegenden Arbeit nur quadratische Jacobi-Matrizen von Interesse. Deshalb werden die bipartiten Graphen G_b in den folgenden Problemdefinitionen so beschränkt, dass deren Knotenmengen V_r und V_c gleich mächtig sind. Die Graphfärbungen werden immer auf bipartiten Graphen betrachtet, weil diese Graphen – im Gegensatz zu den CIG – auch für zweiseitige Färbungen (bidirektionale Partitionen) hinreichend ausdrucksstark sind.

Zuerst wird die NP-Vollständigkeit für die beiden Graphfärbungsprobleme (EDV, ZDV), die zu einer vollständigen Berechnung von Jacobi-Matrizen führen, betrachtet. Im Anschluss daran beschäftigen wir uns mit den beschränkten Graphfärbungsproblemen zur partiellen Berechnung (EDB, ZDB). Zum Schluss werden das EDBD- und das ZDBD-Problem betrachtet.

Da in diesem Kapitel die NP-Vollständigkeit des EDV-, EDB-, ZDV- und ZDB-Problems und damit auch die NP-Schwierigkeit der entsprechenden Minimierungsprobleme gezeigt wird,

werden im folgenden Kapitel keine exakten Verfahren zur Berechnung von minimalen Graphfärbungen, sondern Heuristiken betrachtet.

3.1 Vollständige Färbungen

Zuerst wird die einseitige Färbung betrachtet, die der unidirektionalen Partitionierung einer Jacobi-Matrix J entspricht. Hierbei kann J direkt und vollständig aus der komprimierten Jacobi-Matrix \tilde{J} bestimmt werden. Danach folgt ein Abschnitt über die zweiseitige Graphfärbung (bidirektionale Partitionierung).

3.1.1 Einseitige Färbung

Die einseitige Färbung eines bipartiten Graphen $G_b = (V_r, V_c, E)$ ist eine Möglichkeit, eine unidirektionale Partition einer Jacobi-Matrix J zu berechnen, aus der eine Seed-Matrix S bestimmt werden kann. Es wird eine Färbung gesucht, so dass J direkt und vollständig aus der komprimierten Jacobi-Matrix $\tilde{J} = J \cdot S$ ermittelt werden kann.

Bei der einseitigen Färbung (*einseitige Distanz-2 Färbung von G_b* , vgl. vorheriges Kapitel) sollen entweder die Knoten der Knotenmenge V_c oder die Knoten aus V_r gefärbt werden. Es wird hier nur die Färbung der Knoten aus V_c betrachtet, da man durch ein Vertauschen der Knotenmengen direkt eine Färbung für die Knoten aus V_r berechnen würde. In der Matrix-Terminologie bedeutet dies, dass das Problem der Partitionierung der Zeilen einer Jacobi-Matrix der Partitionierung der Spalten ihrer Transponierten entspricht.

Die *einseitige Distanz-2 Färbung von G_b* wurde schon als Def. 2.4.6 im vorherigen Kapitel eingeführt. Nun wird eine Formalisierung dieser Definition angegeben. Hierbei wird die Färbung als Abbildung Φ eingeführt. Diese gibt an, welchen Knoten welche Farbe zugewiesen wird.

Definition 3.1.1. (*Einseitige Distanz-2 Färbung*) Sei $G_b = (V_r, V_c, E)$ ein bipartiter Graph. Eine Abbildung $\Phi: V_c \rightarrow \{1, \dots, p\}$ ist eine *einseitige Distanz-2 Färbung von G_b bzgl. V_c* , wenn die folgende Bedingung gilt:

1. Für alle Pfade (v, w, x) , $v, x \in V_c$ und $w \in V_r$, ist $\Phi(v) \neq \Phi(x)$.

In [18] ist ein Beweis dafür angegeben, dass das Problem der einseitigen und vollständigen Färbung eines bipartiten Graphen $G_b = (V_r, V_c, E)$ NP-vollständig ist. Im Folgenden werden aber nur diejenigen bipartiten Graphen betrachtet, für die die Bedingung $|V_r| = |V_c|$ gilt. Deshalb wird in diesem Abschnitt ein NP-Vollständigkeitsbeweis für diese Teilklasse der bipartiten Graphen geführt.

Das Minimierungsproblem wurde im vorherigen Kapitel als Problem 2.4.7 vorgestellt. Zunächst wird das zugehörige Entscheidungsproblem 3.1.2 definiert, und dann wird gezeigt, dass

dieses NP-vollständig ist. Daraus kann abgeleitet werden, dass das Minimierungsproblem 2.4.7 NP-schwierig ist.

Problem 3.1.2. (EDV-Problem) Gegeben sei ein bipartiter Graph $G_b = (V_r, V_c, E)$, für den die Bedingung $|V_r| = |V_c|$ gilt, und eine Farbanzahl $p \geq 3$. Existiert eine *einseitige Distanz-2 Färbung* von G_b bzgl. V_c mit p Farben?

Lemma 3.1.3. *Das EDV-Problem ist NP-vollständig.*

Beweis. Zuerst muss gezeigt werden, dass das EDV-Problem \in NP ist: Sei eine Eingabe (G'_b, p') mit $G'_b = (V'_r, V'_c, E')$ des EDV-Problems gegeben. Es wird eine *einseitige Distanz-2 Färbung* Φ' von G'_b bzgl. V'_c geraten und in polynomieller Zeit überprüft, ob die Bedingung aus Def. 3.1.1 dadurch verletzt wird, dass zwei Distanz-2 Nachbarn aus V'_c dieselbe Farbe zugewiesen worden ist.

Dazu wird jeder Knoten $v' \in V'_c$ einmal besucht. Dabei wird überprüft, ob für alle Distanz-2 Nachbarn x' die Bedingung $\Phi'(v') \neq \Phi'(x')$ gilt. Von dem Knoten $v' \in V'_c$ werden also jeweils die Distanz-1 Nachbarn in V'_r und von diesen Nachbarn wiederum die Distanz-1 Nachbarn in V'_c betrachtet.

Somit ist $O(|V'_c| \cdot \bar{\delta}_1(V'_c) \cdot \Delta_1(V'_r)) = O(|E'| \cdot \Delta_1(V'_r))$ eine obere Abschätzung der Laufzeit, wobei $\bar{\delta}_k(V)$ den durchschnittlichen Knotengrad und $\Delta_k(V)$ den maximalen Knotengrad der Knoten aus V bzgl. der Distanz- k Nachbarn bezeichnet. Der Algorithmus ist demnach polynomiell.

Nun wird eine polynomielle Reduktion durchgeführt: Sei $G = (V, E)$ und p eine Eingabe des p -Färbbarkeitsproblems 2.4.4. Diese Eingabe wird in eine Eingabe (G'_b, p') , $G'_b = (V'_r, V'_c, E')$, des EDV-Problems transformiert (vgl. Beispiel in Abb. 3.1.1):

- (T1) $V'_r = \{r'_i : i = 1, \dots, |V|\}$, $V'_c = \{c'_i : i = 1, \dots, |V|\}$,
- (T2) $E' = \{(r'_i, c'_j) : \forall (v_i, v_j) \in E, i, j = 1, \dots, |V|, i < j\} \cup \{(r'_i, c'_i) : i = 1, \dots, |V|\}$,
- (T3) $p' = p$.

Die Transformation ist in polynomieller Zeit möglich: Jeder Knoten $v_i \in V$ wird einmal besucht und es werden jeweils zwei Knoten und eine Kante zu G'_b hinzugefügt. Zudem wird für alle Nachbarknoten $v_j \in V$, $i < j$, eine Kante zu G'_b hinzugefügt, so dass alle Kanten genau einmal betrachtet werden. Somit ist $O(|V| \cdot \frac{1}{2} \bar{\delta}_1(V)) = O(|E|)$ eine obere Schranke des Transformationsalgorithmus.

Nun muss gezeigt werden, dass $(G, p) \in p$ -Färbbarkeitsproblem $\Leftrightarrow (G'_b, p') \in$ EDV-Problem ist; präziser: die Abbildung $\Phi: V \rightarrow \{1, \dots, p\}$ ist genau dann eine *Distanz-1 Färbung* von G , wenn $\Phi': V'_c \rightarrow \{1, \dots, p'\}$ eine *einseitige Distanz-2 Färbung* von G'_b bzgl. V'_c ist, wobei für die Knoten $v_i \in V$ und $c'_i \in V'_c$, $i = 1, \dots, |V|$, die Bedingung $\Phi(v_i) = \Phi'(c'_i)$ gilt.

\Rightarrow : Zu zeigen: Für die Knoten $v_i, v_j \in V$, $i, j = 1, \dots, |V|$, gilt die Bedingung $\Phi(v_i) \neq \Phi(v_j)$ genau dann, wenn für die Knoten $c'_i, c'_j \in V'_c$ die Bedingung $\Phi'(c'_i) \neq \Phi'(c'_j)$ gilt.

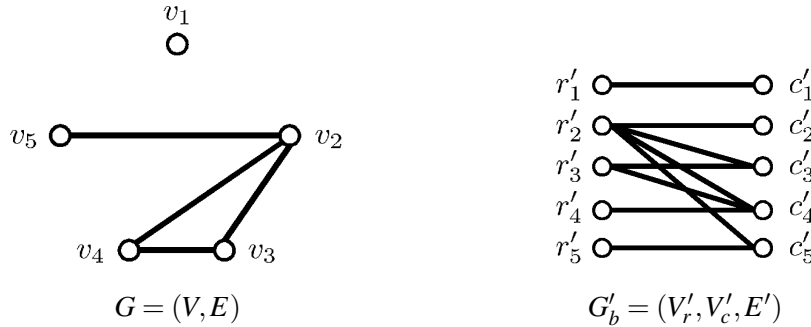


Abbildung 3.1.1: Transformation eines Graphen G der Eingabe des p -Färbbarkeitsproblems 2.4.4 in einen Graphen G'_b der Eingabe des EDV-Problems 3.1.2

Die Knoten $v_i, v_j \in V$ sind nach (T2) genau dann durch eine Kante $(v_i, v_j) \in E$ verbunden, wenn die entsprechenden Knoten $c'_i, c'_j \in V'_c$ über den Pfad (c'_i, r'_i, c'_j) , $r'_i \in V'_r$, oder den Pfad (c'_i, r'_j, c'_j) , $r'_j \in V'_r$, miteinander verbunden sind. Somit gilt: $\Phi(v_i) \neq \Phi(v_j) \Leftrightarrow \Phi'(c'_i) \neq \Phi'(c'_j)$. Ein Beispiel hierfür sind die Knoten $v_2, v_4 \in V$ und $c'_2, c'_4 \in V'_c$ in Abbildung 3.1.1. \square

Damit ist gezeigt, dass das Entscheidungsproblem 3.1.2 NP-vollständig ist. Wenn ein Entscheidungsproblem NP-vollständig ist, dann ist nach [11, Kap. 5.1] das zugehörige Minimierungsproblem NP-schwierig. Daraus folgt, dass das Minimierungsproblem 2.4.7 NP-schwierig ist.

3.1.2 Zweiseitige Färbung

In diesem Abschnitt wird das zweiseitige Färben eines bipartiten Graphen $G_b = (V_r, V_c, E)$ betrachtet. Das Färben entspricht der bidirektionalen Partitionierung einer Jacobi-Matrix J , wenn der bipartite Graph G_b der Dünnsbesetztheitsstruktur dieser Matrix entspricht. Aus so einer zweiseitigen Färbung können zwei Seed-Matrizen S_r und S_c ermittelt werden, so dass eine direkte und vollständige Bestimmung von J aus der komprimierten Jacobi-Matrix $\tilde{J} = S_r \cdot J \cdot S_c$ möglich ist.

Das zweiseitige Graphfärbungsproblem wurde schon im vorherigen Kapitel beschrieben. Es ist hinreichend für die direkte und vollständige Bestimmung von J aus \tilde{J} und geht auf [7, 15] zurück. Die Problemdefinition aus [13] wird zu Grunde gelegt. Zuerst wird die Färbung, das *Star Bicoloring*, definiert. Diese Färbung wird für die nachfolgenden Probleme – das Minimierungs- und das Entscheidungsproblem – benötigt. Zum Schluss wird gezeigt, dass das Entscheidungsproblem NP-vollständig ist und damit das Minimierungsproblem NP-schwierig.

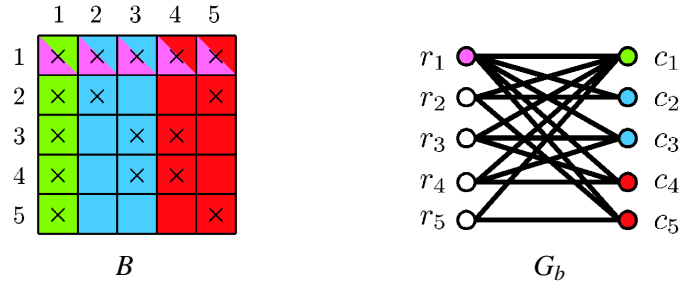


Abbildung 3.1.2: Bipartiter Graph G_b mit *Star Bicoloring* (entsprechendes Dünnbesetztheitsmuster B zum Vergleich identisch gefärbt)

Definition 3.1.4. (*Star Bicoloring*, [13, Def. 5.5]) Sei ein bipartiter Graph $G_b = (V_r, V_c, E)$ gegeben. Die Abbildung $\Phi: [V_r \cup V_c] \rightarrow \{0, 1, \dots, p\}$ ist ein *Star Bicoloring* von G_b , wenn die folgenden Bedingungen gelten:

1. Wenn $v \in V_c$ und $w \in V_r$, dann ist $\Phi(v) \neq \Phi(w)$ oder $\Phi(v) = \Phi(w) = 0$.
2. Wenn $(v, w) \in E$ mit $v \in V_c$ und $w \in V_r$ ist, dann gilt $\Phi(v) \neq 0$ oder $\Phi(w) \neq 0$.
3. Seien die Knoten $v, x \in V_c$ und $w, y \in V_r$:
 - a) Wenn es einen Pfad (v, w, x) mit $\Phi(w) = 0$ gibt, dann muss $\Phi(v) \neq \Phi(x)$ sein.
 - b) Wenn es einen Pfad (w, x, y) mit $\Phi(x) = 0$ gibt, dann muss $\Phi(w) \neq \Phi(y)$ sein.
4. Die vier Knoten auf einem Pfad der Länge drei müssen mit mindestens drei Farben gefärbt werden.

In Abbildung 3.1.2 sehen wir ein Beispiel für ein *Star Bicoloring* Φ von einem bipartiten Graphen G_b . Zum Vergleich wird auch die Matrix B angegeben, die der Dünnbesetztheitsstruktur entspricht, wobei diese identisch gefärbt ist. Kein Knoten aus V_r ist mit derselben Farbe wie ein Knoten aus V_c gefärbt. Des Weiteren ist zu erkennen, dass von jeder Kante mindestens ein Knoten mit einer Farbe $\neq 0$ gefärbt ist. Für jeden Pfad (c_i, r_k, c_j) , $i = 1, \dots, 5$, mit $\Phi(r_k) = 0$ gilt $\Phi(c_i) \neq \Phi(c_j)$, wie z.B. bei dem Pfad (c_1, r_3, c_4) . Die Knoten der Pfade der Länge drei sind mit drei Farben gefärbt, wie z.B. beim Pfad (c_1, r_3, c_3, r_4) .

Nun wird das Minimierungsproblem definiert.

Problem 3.1.5. Sei $G_b = (V_r, V_c, E)$ ein bipartiter Graph mit $|V_r| = |V_c|$. Gesucht wird ein *Star Bicoloring* von G_b mit den wenigsten Farben $p \geq 3$. Vgl. [13, Problem 5.7]

Das entsprechende Entscheidungsproblem lautet wie folgt:

Problem 3.1.6. (ZDV-Problem) Sei $G_b = (V_r, V_c, E)$ ein bipartiter Graph mit $|V_r| = |V_c|$ und p eine Farbanzahl mit $p \geq 3$. Existiert ein *Star Bicoloring* von G_b mit p Farben?

Das ZDV-Problem wird in [7] als *bipartite path p-coloring decision problem* bezeichnet, wobei das *path p-coloring* dem *Star Bicoloring* entspricht. Dort wird weder ein Beweis für die NP-Vollständigkeit des Problems geführt noch ein Verweis auf einen solchen Beweis geliefert. Es wird nur angemerkt, dass die Beweisführung ähnlich zu der für das *cyclic coloring decision*-Problem in [5] sein soll. Die Idee dieses Beweis wird nun im folgenden Lemma verwendet.

Lemma 3.1.7. *Das ZDV-Problem ist NP-vollständig.*

Beweis. Zuerst muss gezeigt werden, dass das ZDV-Problem \in NP ist: Sei eine Eingabe (G'_b, p') , $G'_b = (V'_r, V'_c, E')$, des ZDV-Problems gegeben. Es wird ein *Star Bicoloring* Φ' von G'_b geraten und in polynomieller Zeit überprüft, ob die Knotenfärbung eine Bedingung der Def. 3.1.4 verletzt.

Zur Überprüfung der 1. Bedingung (B1) aus Def. 3.1.4 wird für alle Knoten $v' \in V'_c$ überprüft, ob die Bedingung $\Phi'(v') \neq \Phi'(w')$ für alle $w' \in V'_r$ gilt. Wenn $\Phi'(v') = 0$ ist, dann wird wegen (B2) zusätzlich für alle Distanz-1 Nachbarn w' von v' überprüft, ob die Bedingung $\Phi'(w') \neq 0$ gilt.

Für alle Knoten $v' \in V'_c$ muss nach (B3) überprüft werden, ob die Distanz-2 Nachbarn $x' \in V'_c$ mit einer anderen Farbe als v' gefärbt sind, wenn es einen Pfad (v', w', x') mit $\Phi'(w') = 0$, $w \in V'_r$, gibt. Zudem muss für alle Knoten $w' \in V'_r$ dasselbe für die Distanz-2 Nachbarn $y' \in V'_c$ überprüft werden, wenn es einen Pfad (w', x', y') mit $\Phi'(x') = 0$ gibt.

Zudem werden alle Pfade (v', w', x', y') von $v' \in V'_c$ zu den Distanz-3 Nachbarn $y' \in V'_r$ betrachtet, um wegen (B4) zu überprüfen, ob die vier Knoten dieser Pfade mit mindestens drei Farben gefärbt sind.

Die Überprüfung von (B1) kann mit $O(|V'_c| \cdot |V'_r|)$ und die von (B2) mit $O(|V'_c| \cdot \bar{\delta}_1(V'_c)) = O(|E'|)$ abgeschätzt werden. (B3) kann in $O(|V'_c| \cdot \bar{\delta}_1(V'_c) \cdot \Delta_1(V'_r) + |V'_r| \cdot \bar{\delta}_1(V'_r) \cdot \Delta_1(V'_c)) = O(|E'| \cdot (\Delta_1(V'_r) + \Delta_1(V'_c)))$ und (B4) kann in $O(|V'_c| \cdot \bar{\delta}_1(V'_c) \cdot \Delta_1(V'_r) \cdot \Delta_1(V'_c)) = O(|E'| \cdot \Delta_1(V'_r) \cdot \Delta_1(V'_c))$ überprüft werden. Die Gesamtlaufzeit kann demnach mit $O(|V'_c| \cdot |V'_r| + |E'| \cdot \Delta_1(V'_r) \cdot \Delta_1(V'_c))$ abgeschätzt werden. Damit ist die Überprüfung in polynomieller Zeit möglich.

Nun wird die polynomielle Reduktion gezeigt: Seien der Graph $G = (V, E)$ und p eine Eingabe des p -Färbbarkeitsproblems 2.4.4. Diese Eingabe wird in eine Eingabe (G'_b, p') , $G'_b = (V'_r, V'_c, E')$, des ZDV-Problems transformiert (vgl. Beispiel in Abb. 3.1.3):

- (T1) $V'_r = \{r'_i : i = 1, \dots, |V|\} \cup \{r^{i,j}_1, \dots, r^{i,j}_p : \forall (v_i, v_j) \in E, i, j = 1, \dots, |V|, i < j\}$,
- (T2) $V'_c = \{c'_i : i = 1, \dots, |V|\} \cup \{c^{i,j}_1, \dots, c^{i,j}_p : \forall (v_i, v_j) \in E, i, j = 1, \dots, |V|, i < j\}$,
- (T3) $E' = \{(r'_i, c'_i) : i = 1, \dots, |V|\} \cup \{(r^{i,j}_l, c'_j), (r^{i,j}_l, c^{i,j}_l), (r^{i,j}_l, c'_i) : \forall (v_i, v_j) \in E, i, j = 1, \dots, |V|, i < j, l = 1, \dots, p\}$,
- (T4) $p' = p + 1$.

Für das *Star Bicoloring* wird zusätzlich zu den Farben $1, \dots, p$ die Farbe 0 benötigt. Damit stehen den Eingaben des p -Färbbarkeitsproblems die Farben $1, \dots, p$ und den Eingaben des

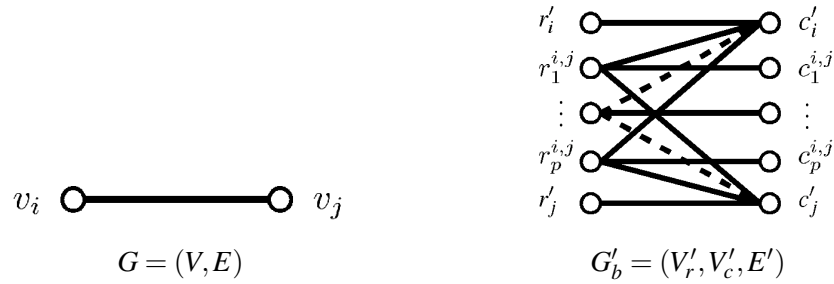


Abbildung 3.1.3: Transformation eines Graphen G der Eingabe des p -Färbbarkeitsproblems 2.4.4 in einen Graphen G'_b der Eingabe des ZDV-Problems 3.1.6

ZDV-Problems die Farben $0, \dots, p$ zur Verfügung.

Die Transformation kann in polynomieller Zeit durchgeführt werden: In Schritt i wird der Knoten $v_i \in V$ besucht. Zwei Knoten und eine Kante werden zum Graphen G'_b hinzugefügt. Für alle zu v_i adjazenten Knoten, die noch nicht besucht wurden, werden $2p$ Knoten und $3p$ Kanten zu G'_b hinzugefügt. Somit ist $O(|V| \cdot \frac{1}{2} \tilde{\delta}_1(V)) = O(|E|)$ eine obere Abschätzung der Laufzeit.

Nun ist zu zeigen, dass $(G, p) \in p$ -Färbbarkeitsproblem $\Leftrightarrow (G'_b, p') \in$ ZDV-Problem ist; präziser: die Abbildung $\Phi: V \rightarrow \{1, \dots, p\}$ ist genau dann eine *Distanz-1 Färbung* von G , wenn $\Phi': [V'_r \cup V'_c] \rightarrow \{0, \dots, p' - 1\}$ ein *Star Bicoloring* von G'_b ist, wobei für die Knoten $v_i \in V$ und $c'_i \in V'_c$, $i = 1, \dots, |V|$, die Bedingung $\Phi(v_i) = \Phi'(c'_i)$ gilt.

\Rightarrow : Sei die Abbildung $\Phi: V \rightarrow \{1, \dots, p\}$ eine *Distanz-1 Färbung* von G . Es ist zu zeigen, dass es ein *Star Bicoloring* $\Phi': [V'_r \cup V'_c] \rightarrow \{0, \dots, p' - 1\}$ von G'_b gibt, so dass die Bedingung $\Phi(v_i) = \Phi'(c'_i)$, $i = 1, \dots, |V|$, gilt.

Wenn es eine Kante $(v_i, v_j) \in E$ mit $v_i, v_j \in V$, $i, j = 1, \dots, |V|$, gibt, gilt nach Bedingung 1 der Def. 3.1.1 $\Phi(v_i) \neq \Phi(v_j)$. Somit müssen die entsprechenden Knoten $c'_i, c'_j \in V'_c$ durch Φ' unterschiedlich gefärbt werden. Eine Möglichkeit, die Knoten aus $V'_r \cup V'_c$ zu färben, ist die Folgende: Allen Knoten aus V'_r wird die Farbe 0 zugewiesen. Die Knoten aus V'_c werden so gefärbt, dass die Bedingungen $\Phi'(c_l^{i,j}) \neq \Phi'(c'_i)$, $\Phi'(c_l^{i,j}) \neq \Phi'(c'_j)$ und $\Phi'(c_l^{i,j}) \neq 0$ gelten, wobei $(v_i, v_j) \in E$ und $l = 1, \dots, p$ ist. Das bedeutet nichts anderes als, dass die Distanz-2 Nachbarn c'_i und c'_j jeweils mit einer anderen Farbe gefärbt werden und den Knoten $c_l^{i,j}$, $l = 1, \dots, p$, eine weitere Farbe zugewiesen wird. Dafür gibt es hinreichend viele Farben, da $p' \geq 4$ ist wegen $p \geq 3$. Somit ist die Abbildung Φ' ein *Star Bicoloring* von G'_b .

Damit ist gezeigt worden, dass es zu jeder *Distanz-1 Färbung* Φ immer ein passendes *Star Bicoloring* Φ' gibt.

\Leftarrow : Zu Zeigen: Wenn es ein *Star Bicoloring* Φ' von G'_b gibt, dann gibt es auch eine *Distanz-1 Färbung* Φ von G , so dass $\Phi(v_i) = \Phi'(c'_i)$, $i = 1, \dots, |V| = |V_c|$, gilt.

Damit die Abbildung Φ eine *Distanz-1 Färbung* von G_b ist, darf kein Knoten $v_i \in V$, $i = 1, \dots, |V|$, mit der Farbe 0 gefärbt werden. Deshalb dürfen die entsprechenden Knoten $c'_i \in V'_c$ auch nicht mit der Farbe 0 gefärbt sein. Sei nun $\Phi'(c'_i) = 0$. Dann müssten wegen der 3. Bedingung (B3) in Def. 3.1.4 alle p' Nachbarknoten paarweise unterschiedlich gefärbt werden. Dazu würden $p' + 1$ Farben benötigt. Das ist ein Widerspruch, da nur p' Farben vorhanden sind. Somit ist der Knoten c'_i nie mit der Farbe 0 gefärbt und es gilt immer $\Phi(v_i) \neq 0$.

Des Weiteren müssen die Knoten $v_i, v_j \in V$, $i, j = 1, \dots, |V|$, unterschiedlich gefärbt werden, wenn eine Kante $(v_i, v_j) \in E$ existiert. Somit muss auch für die entsprechenden Knoten $c'_i, c'_j \in V'_c$ die Bedingung $\Phi'(c'_i) \neq \Phi'(c'_j)$ gelten. Sei nun $\Phi'(c'_i) = \Phi'(c'_j)$. In diesem Fall müssten die Knoten $r_1^{i,j}, \dots, r_p^{i,j}$ und r'_i wegen (B4) in Def. 3.1.4 paarweise unterschiedlich gefärbt werden, damit die Knoten, die auf einem Pfad $(r'_i, c'_i, r_l^{i,j}, c'_j)$, $l = 1, \dots, p$, liegen, mit mindestens drei unterschiedlichen Farben gefärbt sind. Dies führt zu einem Widerspruch, da dafür mindestens $p' + 1$ Farben benötigt würden. Somit wäre die Abbildung Φ' kein *Star Bicoloring* mit p' Farben. Deshalb muss die Bedingung $\Phi'(c'_i) \neq \Phi'(c'_j)$ gelten. Daraus folgt direkt, dass auch $\Phi(v_i) \neq \Phi(v_j)$ immer gilt.

Abschließend muss noch der Sonderfall betrachtet werden, bei dem es eine Zusammenhangskomponente mit nur einem einzigen Knoten gibt: Für einen solchen Knoten $v_i \in V$, $i = 1, \dots, |V|$, existiert wegen der Transformation eine Zusammenhangskomponente in G'_b mit den adjazenten Knoten c'_i und r'_i . Nun sei $\Phi'(r'_i) \neq 0$ und $\Phi'(c'_i) = 0$, dann müsste auch $\Phi(v_i) = 0$ sein. Dieses darf aber nicht sein. O.B.d.A wird deshalb angenommen, dass in diesem Fall $\Phi(r'_i) = 0$ ist und somit $\Phi(c'_i) \neq 0$ ist. \square

Damit ist gezeigt, dass das Entscheidungsproblem 3.1.6 NP-vollständig ist. Somit ist das Minimierungsproblem 3.1.5 NP-schwierig.

3.2 Beschränkte Färbungen

Im vorherigen Abschnitt sind die Probleme der einseitigen und zweiseitigen Graphfärbungen betrachtet worden, deren Resultate zu einer Seed-Matrix bzw. zwei Seed-Matrizen führen, so dass aus der komprimierten Jacobi-Matrix \tilde{J} alle Elemente der Jacobi-Matrix J direkt bestimmt werden können. Im Folgenden muss allerdings nur noch eine Teilmenge der Elemente direkt bestimmt werden können. Somit können die Färbungen auf diese Teilmenge bzw. die entsprechenden Kanten des passenden bipartiten Graphen beschränkt werden. Oftmals können so noch mehr Richtungsableitungen als eine Linearkombination berechnet werden, da die nicht benötigten Elemente sich gegenseitig beeinflussen dürfen (vgl. Abschnitt 2.5). Die Graphfärbung wird entsprechend angepasst, so dass oftmals weniger Farben benötigt werden.

Die zwei vorgestellten Probleme für die vollständige Bestimmung von Jacobi-Matrizen werden auf die partielle Berechnung angepasst, indem die Färbungen beschränkt werden. Dieser

Abschnitt ist in die Unterabschnitte *Beschränkte einseitige Färbung* und *Beschränkte zweiseitige Färbung* unterteilt. Zuerst wird jeweils die Definition der modifizierten Färbung, dann werden die Probleme und zum Schluss der NP-Vollständigkeitsbeweis angegeben.

Abschließend wird der Fall betrachtet, in dem die Menge der benötigten Elemente die Hauptdiagonalelemente enthält.

3.2.1 Beschränkte einseitige Färbung

Beim EDV-Problem 3.1.2 können alle Elemente der Jacobi-Matrix J direkt aus der komprimierten Jacobi-Matrix \tilde{J} bestimmt werden. Beim EDB-Problem (einseitige und beschränkte Färbung zur direkten und partiellen Bestimmung von J) müssen nur diejenigen Elemente, die in der Menge der benötigten Elemente R enthalten sind, direkt bestimmt werden können.

Alle Spalten von J , die paarweise in denselben Komponenten weder zwei benötigte Elemente noch ein benötigtes und ein nicht benötigtes Element enthalten, dürfen nun gleichzeitig berechnet werden. Hier müssen die Spalten demnach nicht strukturell orthogonal sein.

Die *einseitige Distanz-2 Färbung* von G_b (Def. 2.4.6), die für das EDV-Problem hinreichend ist, ist die Grundlage für die folgende Färbung. Diese wird auf die Menge E_R , die diejenigen Kanten enthält, die den benötigten Elementen entsprechen, beschränkt, so dass alle Knoten in V_c genau dann unterschiedlich gefärbt werden, wenn sie über einen Pfad der Länge zwei miteinander verbunden sind, von dem mindestens eine Kante in E_R enthalten ist.

Definition 3.2.1. (*Beschränkte einseitige Distanz-2 Färbung*, vgl. [13, Abschnitt 8.1]) Sei $G_b = (V_r, V_c, E)$ ein bipartiter Graph. Die Menge $E_R \subseteq E$ enthalte die Kanten, die den benötigten Elementen entsprechen. Eine Abbildung $\Phi: V_c \rightarrow \{0, 1, \dots, p\}$ ist eine *einseitige Distanz-2 Färbung* von G_b bzgl. V_c beschränkt auf E_R , wenn die folgenden Bedingungen gelten:

1. Für alle Kanten $(v, w) \in E_R$, $v \in V_c$ und $w \in V_r$, ist $\Phi(v) \neq 0$.
2. Für alle Pfade (v, w, x) , $v, x \in V_c$, $w \in V_r$ und $(v, w) \in E_R$, ist $\Phi(v) \neq \Phi(x)$.

Die Farbe 0 kann also Knoten zugewiesen werden, die nicht inzident zu einer Kante aus E_R sind. Solche Knoten sind somit zu keiner Kante inzident, die einem benötigten Element entspricht.

Nun wird zuerst das Minimierungsproblem und danach das EDB-Entscheidungsproblem definiert.

Problem 3.2.2. Sei $G_b = (V_r, V_c, E)$ ein bipartiter Graph mit $|V_r| = |V_c|$. Zusätzlich sei die Menge E_R gegeben, die die Kanten, die den benötigten Elementen entsprechen, enthält. Gesucht wird eine *einseitige Distanz-2 Färbung* von G_b bzgl. V_c beschränkt auf E_R mit den wenigsten Farben $p \geq 3$.

Problem 3.2.3. (EDB-Problem) Sei $G_b = (V_r, V_c, E)$ ein bipartiter Graph mit $|V_r| = |V_c|$ und p eine Farbanzahl mit $p \geq 3$. Zusätzlich sei die Menge E_R gegeben, die die Kanten, die den benötigten Elementen entsprechen, enthält. Existiert eine *einseitige Distanz-2 Färbung* von G_b bzgl. V_c beschränkt auf E_R mit p Farben?

Das EDB-Problem ist eine Oberklasse des EDV-Problems 3.1.2. Die NP-Schwierigkeit des EDB-Problems wird nun durch eine Reduktion vom EDV-Problem auf das EDB-Problem gezeigt. Der entscheidende Punkt der Transformation ist der, dass die Menge E_R , die Teil der Eingabe des EDB-Problems ist, alle Kanten aus E enthält.

Lemma 3.2.4. *Das EDB-Problem ist NP-vollständig.*

Beweis. Zuerst muss gezeigt werden, dass das EDB-Problem \in NP ist: Gegeben sei eine Eingabe (G_b^*, E_R^*, p^*) , $G_b^* = (V_r^*, V_c^*, E^*)$, des EDB-Problems. Eine *einseitige Distanz-2 Färbung* Φ^* von G_b^* bzgl. V_c^* beschränkt auf E_R^* wird geraten. Dann wird in polynomieller Zeit überprüft, ob die Knotenfärbung eine Bedingung der Def. 3.2.1 verletzt.

In jedem Schritt wird ein Knoten $v^* \in V_c^*$ betrachtet. Dabei wird für alle Distanz-1 Nachbarn $w^* \in V_r^*$ von v^* mit $(v^*, w^*) \in E_R^*$ wegen Bedingung 1 (B1) aus Def. 3.2.1 überprüft, ob die Bedingung $\Phi^*(v^*) \neq 0$ gilt. Zusätzlich wird für alle Distanz-2 Nachbarn $x^* \in V_c^*$, die mit dem Knoten v^* über einen Pfad mit mindestens einer Kante aus E_R^* verbunden sind, wegen (B2) aus Def. 3.2.1 überprüft, ob $\Phi^*(v^*) \neq \Phi^*(x^*)$ gilt.

Die Überprüfung der ersten Bedingung kann in $O(|V_c^*| \cdot \bar{\delta}_1(V_c^*)) = O(|E^*|)$ und der zweiten Bedingung in $O(|V_c^*| \cdot \bar{\delta}_1(V_c^*) \cdot \Delta_1(V_r^*)) = O(|E^*| \cdot \Delta_1(V_r^*))$ vorgenommen werden. Somit ist $O(|E^*| \cdot \Delta_1(V_r^*))$ eine obere Schranke. Damit ist gezeigt, dass der Algorithmus polynomiell ist.

Nun wird die polynomielle Reduktion durchgeführt. Sei $G'_b = (V'_r, V'_c, E')$ und p' eine Eingabe des EDV-Problems 3.1.2. Diese Eingabe wird in eine Eingabe (G_b^*, E_R^*, p^*) , $G_b^* = (V_r^*, V_c^*, E^*)$, des EDB-Problems transformiert (vgl. Beispiel in Abb. 3.2.1):

- (T1) $G_b^* = (V_r^*, V_c^*, E^*)$, wobei $V_r^* = V'_r$, $V_c^* = V'_c$ und $E^* = E'$,
- (T2) $E_R^* = E'$,
- (T3) $p^* = p' + 1$.

Für die *einseitige Distanz-2 Färbung* von G_b^* bzgl. V_c^* beschränkt auf E_R^* wird zusätzlich zu den Farben $1, \dots, p'$ die Farbe 0 benötigt. Mit dieser Farbe werden diejenigen Knoten gefärbt, die nicht inzident zu einer Kante aus E_R sind. Damit stehen den Eingaben des EDV-Problems die Farben $1, \dots, p'$ und den Eingaben des EDB-Problems die Farben $0, \dots, p'$ zur Verfügung.

Die Transformation ist offensichtlich in polynomieller Zeit möglich: Da der Graph G'_b und der konstruierte Graph G_b^* nach der Transformation identisch sind, reicht es aus, jede Kante von G'_b einmal zu besuchen und jeweils einen Knoten zu V_r^* , einen Knoten zu V_c^* und eine Kante zu E^* und E_R^* hinzuzufügen. Die Laufzeit kann demnach mit $O(|E^*|)$ abgeschätzt werden.

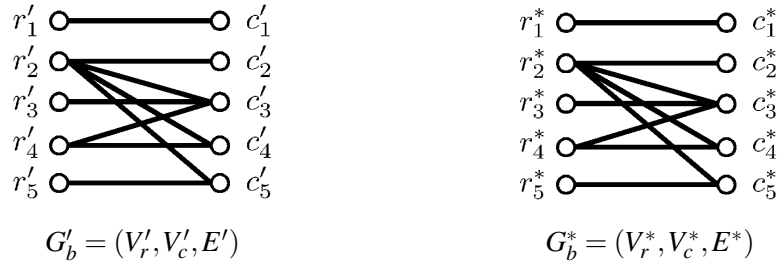


Abbildung 3.2.1: Transformation eines Graphen G'_b der Eingabe des EDV-Problems 3.1.2 in einen Graphen G_b^* der Eingabe des EDB-Problems 3.2.3

Nun bleibt zu zeigen, dass $(G'_b, p') \in \text{EDV-Problem} \Leftrightarrow (G_b^*, E_R^*, p^*) \in \text{EDB-Problem}$; präzisier: die Abbildung $\Phi': V'_c \rightarrow \{1, \dots, p'\}$ ist genau dann eine *einseitige Distanz-2 Färbung* von G'_b bzgl. V'_c , wenn $\Phi^*: V_c^* \rightarrow \{0, \dots, p^* - 1\}$ eine *einseitige Distanz-2 Färbung* von G_b^* bzgl. V_c^* beschränkt auf E_R^* ist, wobei für die Knoten $c'_i \in V'_c$ und $c_i^* \in V_c^*$, $i = 1, \dots, |V'_c|$, die Bedingung $\Phi'(c'_i) = \Phi^*(c_i^*)$ gilt.

\Rightarrow : Die Knoten $c'_i, c'_j \in V'_c$, $i, j = 1, \dots, |V'_c|$, sind nach Bedingung 1 (B1) der Definition 3.1.1 genau dann über einen Pfad (c'_i, r'_k, c'_j) der Länge zwei mit $r'_k \in V'_r$ verbunden, wenn die Knoten $c_i^*, c_j^* \in V_c^*$ nach (B2) in Def. 3.2.1 über den Pfad (c_i^*, r_k^*, c_j^*) , $r_k^* \in V_r^*$, miteinander verbunden sind, weil nach (T2) die Kanten (c_i^*, r_k^*) und (r_k^*, c_j^*) aus E_R^* sind. Somit gilt: $\Phi'(c'_i) \neq \Phi'(c'_j) \Leftrightarrow \Phi^*(c_i^*) \neq \Phi^*(c_j^*)$. Als Beispiel betrachten wir Abb. 3.2.1. Die Knoten $c'_3, c'_4 \in V'_c$, verbunden über $r'_4 \in V'_r$, und die Knoten $c_3^*, c_4^* \in V_c^*$, verbunden über $r_4^* \in V_r^*$, müssen unterschiedlich gefärbt werden.

Zudem darf keinem Knoten $c_i^* \in V_c^*$, $i = 1, \dots, |V_c^*|$, wegen (B2) in Def. 3.2.1 die Farbe 0 zugewiesen werden, da nach (T2) jeder Knoten inzident zu einer Kante aus E_R^* ist. Dies ist entscheidend, da der Abbildung Φ' die Farbe 0 nicht zur Verfügung steht.

Die Knoten in G'_b müssen also genau dann unterschiedlich gefärbt werden, wenn die entsprechenden Knoten in G_b^* unterschiedlich gefärbt werden müssen. Weil zudem keinem Knoten in G_b^* die Farbe 0 zugewiesen werden darf, ist die Abbildung Φ' genau dann eine *einseitige Distanz-2 Färbung* von G'_b bzgl. V'_c , wenn Φ^* eine *einseitige Distanz-2 Färbung* von G_b^* bzgl. V_c^* beschränkt auf E_R^* ist, wobei $\Phi'(c'_i) = \Phi^*(c_i^*)$, $i = 1, \dots, |V'_c|$, gilt. \square

Somit ist das Entscheidungsproblem NP-vollständig und damit das entsprechende Minimierungsproblem 3.2.2 NP-schwierig.

3.2.2 Beschränkte zweiseitige Färbung

Im Folgenden wird eine zweiseitige Färbung betrachtet, die eine direkte und partielle Berechnung einer Jacobi-Matrix J ermöglicht. Zusätzlich zu der Eingabe bei der vollständigen Be-

rechnung ist die Menge E_R erforderlich, die die Kanten enthält, die den benötigten Elementen entsprechen.

Das *Star Bicoloring* von G_b (Def. 3.1.4) muss zur beschränkten Färbung so angepasst werden, dass die Menge E_R in adäquater Weise berücksichtigt wird. Dieser Abschnitt ist wie die Vorherigen aufgebaut: Zuerst werden die Färbung, das Minimierungs- und das Entscheidungsproblem definiert und anschließend der entsprechende NP-Vollständigkeitsbeweis geführt.

Definition 3.2.5. (Beschränktes Star Bicoloring, vgl. [13, Abschnitt 8.3]) Sei ein bipartiter Graph $G_b = (V_r, V_c, E)$ gegeben. Die Menge $E_R \subseteq E$ enthalte die Kanten, die den benötigten Elementen entsprechen. Eine Abbildung $\Phi: [V_r \cup V_c] \rightarrow \{0, 1, \dots, p\}$ ist ein *Star Bicoloring* von G_b beschränkt auf E_R , wenn die folgenden Bedingungen gelten:

1. Wenn $v \in V_c$ und $w \in V_r$, dann ist $\Phi(v) \neq \Phi(w)$ oder $\Phi(v) = \Phi(w) = 0$.
2. Wenn $(v, w) \in E_R$ mit $v \in V_c$ und $w \in V_r$, dann ist $\Phi(v) \neq 0$ oder $\Phi(w) \neq 0$.
3. Seien die Knoten $v, x \in V_c$, $w, y \in V_r$ und die Kante $(w, x) \in E_R$:
 - a) Wenn es einen Pfad (v, w, x) mit $\Phi(w) = 0$ gibt, ist $\Phi(v) \neq \Phi(x)$.
 - b) Wenn es einen Pfad (w, x, y) mit $\Phi(x) = 0$ gibt, ist $\Phi(w) \neq \Phi(y)$.
4. Wenn es einen Pfad (v, w, x, y) , $v, x \in V_c$ und $w, y \in V_r$, mit $\Phi(w) \neq 0$ und $\Phi(x) \neq 0$ gibt, dann ist $\Phi(v) \neq \Phi(x)$ oder $\Phi(w) \neq \Phi(y)$.

In Abbildung 3.2.2 ist ein Beispiel für ein *Beschränktes Star Bicoloring* eines bipartiten Graphen G_b angeführt. In der entsprechenden, identisch gefärbten Dünnbesetztheitsstruktur (Matrix B) können die Bedingungen nachvollzogen werden. Von jeder Kante aus E_R muss mindestens ein Knoten mit einer Farbe $\neq 0$ gefärbt sein, da das entsprechende Element aus B ansonsten nicht direkt bestimmt werden kann. Nach der 3. Bedingung der Def. 3.2.5 ist z.B. der Pfad (c_1, r_2, c_2) korrekt gefärbt. Wäre $\Phi(c_1) = \Phi(c_2)$, dann könnten die Elemente der Jacobi-Matrix J , die den Elemente $b_{1,1}$ und $b_{2,2}$ der Matrix B entsprechen, nicht mehr direkt bestimmt werden. Als Beispiel für die 4. Bedingung betrachten wir den Pfad (r_4, c_3, r_3, c_4) . Wären die Knoten r_3 und r_4 nicht unterschiedlich gefärbt, dann könnten die Werte der Elemente $b_{3,3}$ und $b_{4,4}$ nicht direkt bestimmt werden.

Problem 3.2.6. Gegeben sei ein bipartiter Graph $G_b = (V_r, V_c, E)$ mit $|V_r| = |V_c|$. Zusätzlich sei die Menge E_R gegeben, die die Kanten enthält, die den benötigten Elementen entsprechen. Gesucht wird ein *Star Bicoloring* von G_b beschränkt auf E_R , mit den wenigsten Farben $p \geq 3$.

Das entsprechende Entscheidungsproblem ist wie folgt definiert:

Problem 3.2.7. (ZDB-Problem) Sei $G_b = (V_r, V_c, E)$ ein bipartiter Graph mit $|V_r| = |V_c|$ und p eine Farbanzahl mit $p \geq 3$. Zusätzlich sei die Menge E_R gegeben, die die Kanten enthält, die den benötigten Elementen entsprechen. Existiert ein *Star Bicoloring* von G_b beschränkt auf E_R mit p Farben?

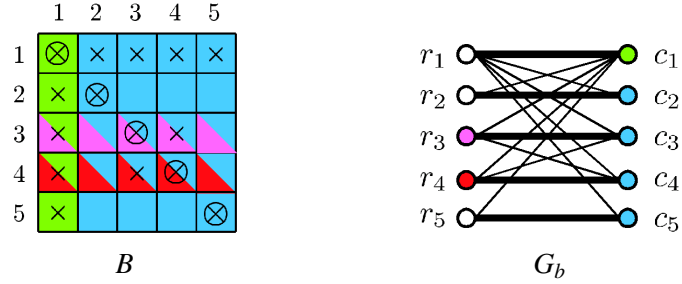


Abbildung 3.2.2: Bipartiter Graph G_b mit *Beschränktem Star Bicoloring*; passende Matrix B (entspricht Dünnesetztheitsstruktur von J) zum Vergleich identisch gefärbt

Das ZDB-Problem ist eine Oberklasse des ZDV-Problems 3.1.6. Die NP-Schwierigkeit des ZDB-Problems wird im Folgenden durch eine Reduktion von Problem 3.1.6 gezeigt. Der entscheidende Punkt ist wieder – wie bei der Reduktion auf das EDB-Problem – die Wahl von E_R : E_R enthält alle Kanten aus E .

Lemma 3.2.8. *Das ZDB-Problem ist NP-vollständig.*

Beweis. Zuerst muss gezeigt werden, dass das ZDB-Problem \in NP ist: Gegeben sei eine Eingabe (G_b^*, E_R^*, p^*) , $G_b^* = (V_r^*, V_c^*, E^*)$, des ZDB-Problems. Ein *Star Bicoloring* Φ^* von G_b^* beschränkt auf E_R^* wird geraten. In polynomieller Zeit muss überprüft werden können, ob die Färbung eine Bedingung der Def. 3.2.5 verletzt.

Für jedes $v^* \in V_c^*$ wird wegen Bedingung 1 (B1) aus Definition 3.2.5 überprüft, ob für alle Distanz-1 Nachbarn $w^* \in V_r^*$ eines Knotens v^* die Bedingung $\Phi^*(v^*) \neq \Phi^*(w^*)$ gilt. Wenn für einen Knoten v^* die Bedingung $\Phi^*(v^*) = 0$ gilt, dann muss nach (B2) für alle Distanz-1 Nachbarn w^* , die über eine Kante aus E_R^* mit v^* verbunden sind, überprüft werden, ob die Bedingung $\Phi^*(w^*) \neq 0$ gilt.

Für (B3) muss überprüft werden, ob die Distanz-2 Nachbarn $x^* \in V_c^*$ von $v^* \in V_c^*$ mit einer anderen Farbe als v^* gefärbt sind, wenn es einen Pfad (v^*, w^*, x^*) , $w^* \in V_r^*$, mit $(w^*, x^*) \in E_R^*$ und $\Phi^*(w^*) = 0$ gibt. Zudem muss für die Knoten $w^* \in V_r^*$ die entsprechende Überprüfung für die Pfade (w^*, x^*, y^*) mit $(w^*, x^*) \in E_R^*$ und $\Phi^*(x^*) = 0$ durchgeführt werden.

Des Weiteren müssen alle Pfade (v^*, w^*, x^*, y^*) , $v^*, x^* \in V_c^*$ und $w^*, y^* \in V_r^*$, für die die Bedingungen $(w^*, x^*) \in E_R^*$, $\Phi^*(w^*) \neq 0$ und $\Phi^*(x^*) \neq 0$ gelten, zu den Distanz-3 Nachbarn y^* von v^* betrachtet werden, um wegen (B4) zu überprüfen, ob die Bedingung $\Phi^*(v^*) \neq \Phi^*(x^*)$ oder $\Phi^*(w^*) \neq \Phi^*(y^*)$ gilt.

Die obere Schranke für die Laufzeit des Algorithmus in Lemma 3.1.7, der überprüft, ob das Problem aus der Klasse NP ist, ist auch hier adäquat. Wegen der Beschränkung durch die Menge E_R^* müssen nicht mehr Distanz-1, -2 und -3 Nachbarn betrachtet werden. Somit ist $O(|V_r^*| \cdot |V_c^*| + |E^*| \cdot \Delta_1(V_r^*) \cdot \Delta_1(V_c^*))$ eine obere Schranke der Laufzeit. Daraus folgt, dass der

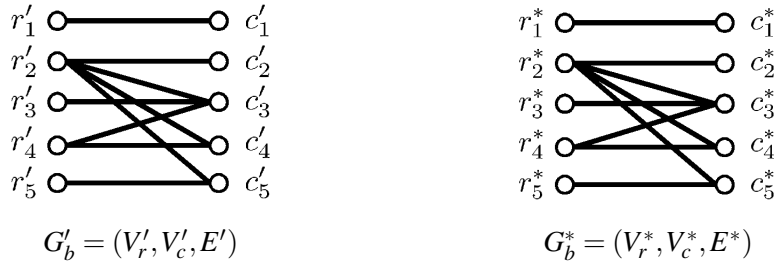


Abbildung 3.2.3: Transformation eines Graphen G'_b der Eingabe des ZDV-Problems 3.1.6 in einen Graphen G_b^* der Eingabe des ZDB-Problems 3.2.7

Algorithmus polynomiell ist.

Nun kommen wir zur polynomiellen Reduktion: Sei $G'_b = (V'_r, V'_c, E')$ und p' eine Eingabe des ZDV-Problems 3.1.6. Durch die folgende Transformation wird eine Eingabe (G_b^*, E_R^*, p^*) , $G_b^* = (V_r^*, V_c^*, E^*)$, des ZDB-Problems konstruiert (vgl. Beispiel in Abb. 3.2.3):

- (T1) $G_b^* = (V_r^*, V_c^*, E^*)$, wobei $V_r^* = V'_r, V_c^* = V'_c$ und $E^* = E'$,
- (T2) $E_R^* = E'$,
- (T3) $p^* = p'$.

Die Transformation ist in polynomieller Zeit möglich: Da der Graph G'_b und der konstruierte Graph G_b^* identisch sind, reicht es aus, jede Kante von G'_b einmal zu besuchen und einen Knoten zu V_r^* , einen Knoten zu V_c^* und eine Kante zu E^* und E_R^* hinzuzufügen. Der Aufwand kann demnach mit $O(|E^*|)$ abgeschätzt werden.

Nun bleibt zu zeigen, dass $(G'_b, p') \in \text{ZDV-Problem} \Leftrightarrow (G_b^*, E_R^*, p^*) \in \text{ZDB-Problem}$; präziser: die Abbildung $\Phi': [V'_r \cup V'_c] \rightarrow \{0, \dots, p' - 1\}$ ist genau dann ein *Star Bicoloring* von G'_b , wenn $\Phi^*: [V_r^* \cup V_c^*] \rightarrow \{0, \dots, p^* - 1\}$ ein *Star Bicoloring* von G_b^* beschränkt auf E_R^* ist, wobei für die Knoten $r'_i, c'_i \in V'_c$ und $r_i^*, c_i^* \in V_c^*$, $i = 1, \dots, |V'_r|$, die Bedingungen $\Phi'(r'_i) = \Phi^*(r_i^*)$ und $\Phi'(c'_i) = \Phi^*(c_i^*)$ gelten.

\Leftarrow : Für die Knoten $r'_i \in V'_r$ und $c'_j \in V'_c$, $i, j = 1, \dots, |V'_r|$, gilt nach Bedingung 1 (B1) der Def. 3.1.4 die Bedingung $\Phi'(r'_i) \neq \Phi'(c'_j)$ genau dann, wenn für die Knoten $r_i^* \in V_r^*$ und $c_j^* \in V_c^*$, $i, j = 1, \dots, |V_r^*|$ wegen (B1) aus Def. 3.2.5 die Bedingung $\Phi^*(r_i^*) \neq \Phi^*(c_j^*)$ gilt, wobei $\Phi'(r'_i) \neq 0$ oder $\Phi'(c'_j) \neq 0$ und $\Phi^*(r_i^*) \neq 0$ oder $\Phi^*(c_j^*) \neq 0$ vorausgesetzt wird. Nur mit der Farbe 0 dürfen sowohl Knoten aus der Knotenmenge V'_r bzw. V_r^* als auch Knoten aus der Knotenmenge V'_c bzw. V_c^* gefärbt werden, da die Farbe 0 für keinen AD-Berechnungsdurchlauf steht.

Nun werden alle Kanten $(r'_i, c'_j) \in E'$, $r'_i \in V'_r, c'_j \in V'_c$ und $i, j = 1, \dots, |V'_r|$, und die entsprechenden Kanten $(r_i^*, c_j^*) \in E^*$, $r_i^* \in V_r^*, c_j^* \in V_c^*$, betrachtet. Sei nun $\Phi'(r'_i) = \Phi^*(r_i^*) = 0$. Dann gilt die Bedingung $\Phi'(c'_j) \neq 0$ wegen (B2) aus Def. 3.1.4 genau dann, wenn wegen (B2) aus Def. 3.2.5 die Bedingung $\Phi^*(c_j^*) \neq 0$ gilt, da E_R^* alle Kanten aus E^* enthält. Analog dazu der

Fall: $\Phi'(c'_i) = \Phi^*(c_i^*) = 0$.

Jetzt betrachten wir die Pfade (v', w', x') mit $v', x' \in V'_c$, $w' \in V'_r$ und $\Phi'(w') = 0$ und die entsprechenden Pfade (v^*, w^*, x^*) mit $v^*, x^* \in V_c^*$, $w^* \in V_r^*$ und $\Phi^*(w^*) = 0$. Dann müssen die Knoten v' und x' nach (B3) aus Def. 3.1.4 genau dann unterschiedlich gefärbt werden, wenn $\Phi^*(v^*) \neq \Phi^*(x^*)$ nach (B3) aus Def. 3.2.5 gilt, da $(w^*, x^*) \in E_R^*$ ist. Analog dazu müssen die Pfade (w', x', y') mit $w', y' \in V'_r$ und $x' \in V'_c$ und die entsprechenden Pfade (w^*, x^*, y^*) mit $w^*, y^* \in V_r^*$ und $x^* \in V_c^*$ betrachtet werden, wobei $\Phi'(x') = \Phi^*(x^*) = 0$ gilt.

Die Pfade (v', w', x', y') , $v', x' \in V'_c$ und $w', y' \in V'_r$, und die entspr. Pfade (v^*, w^*, x^*, y^*) , $v^*, x^* \in V_c^*$ und $w^*, y^* \in V_r^*$, werden betrachtet, wobei jede Kante $(w^*, x^*) \in E_R^*$ ist, da $E_R^* = E^*$ gilt. Nun muss gezeigt werden, dass die Knoten der entsprechenden Pfade immer mit drei Farben gefärbt sein müssen, um (B4) aus Def. 3.1.4 zu erfüllen. Zudem muss gezeigt werden, dass unter der Voraussetzung $\Phi^*(w^*) \neq 0$ und $\Phi^*(x^*) \neq 0$ bzw. $\Phi'(w') \neq 0$ und $\Phi'(x') \neq 0$ wegen (B4) aus Def. 3.1.4 eine der Bedingungen $\Phi'(v') \neq \Phi'(x')$ oder $\Phi'(w') \neq \Phi'(y')$ (1. Pfad) genau dann gilt, wenn wegen (B4) aus Def. 3.2.5 die Bedingung $\Phi^*(v^*) \neq \Phi^*(x^*)$ oder die Bedingung $\Phi^*(w^*) \neq \Phi^*(y^*)$ (2. Pfad) gilt. Dazu werden die zwei folgenden Punkte betrachtet:

1. Es sei $\Phi'(w') = \Phi^*(w^*) = 0$. Dann gelten nach (B3) aus Def. 3.1.4 die Bedingungen $\Phi'(v') \neq \Phi'(x')$ und $\Phi^*(v^*) \neq \Phi^*(x^*)$, da $(v^*, w^*) \in E_R^*$ ist. Zudem gilt nach (B3) aus Def. 3.2.5, dass $\Phi'(v') \neq 0$ und $\Phi'(x') \neq 0$ sind, da (v', w') , (w', x') in E' sind. Entsprechend sind $\Phi^*(v^*) \neq 0$ und $\Phi^*(x^*) \neq 0$, da alle Kanten in E_R^* enthalten sind. Damit sind die Pfade (v', w', x', y') und (v^*, w^*, x^*, y^*) jeweils mit mindestens drei Farben gefärbt. Analog dazu: $\Phi'(x') = \Phi^*(x^*) = 0$.

2. Seien $\Phi'(w') = \Phi^*(w^*) \neq 0$ und $\Phi'(x') = \Phi^*(x^*) \neq 0$. Dann muss $\Phi'(v') \neq \Phi'(x')$ oder $\Phi'(w') \neq \Phi'(y')$ gelten um (B4) aus Def. 3.1.4 zu erfüllen. Dementsprechend muss wegen (B4) aus Def. 3.2.5 eine der Bedingungen $\Phi^*(v^*) \neq \Phi^*(x^*)$ oder $\Phi^*(w^*) \neq \Phi^*(y^*)$ gelten, weil $(w^*, x^*) \in E_R^*$ ist. Die Knoten der beiden Pfade sind zudem jeweils mit drei Farben gefärbt.

Damit ist gezeigt, dass die Abbildung Φ' genau dann ein *Star Bicoloring* von G'_b ist, wenn die Abbildung Φ^* ein *Star Bicoloring* von G_b^* beschränkt auf E_R^* ist. \square

Somit ist das Minimierungsproblem 3.2.6 NP-schwierig.

3.2.3 Färbungen zur partiellen Berechnung der Hauptdiagonalelemente

In diesem Abschnitt sollen nur die Hauptdiagonalelemente einer Jacobi-Matrix J partiell berechnet werden. Dieses Problem kann als EDB- oder ZDB-Problem betrachtet werden, wobei $G_b = (V_r, V_c, E)$ den bipartiten Graphen bezeichnet und die Menge E_R die Kanten enthält, die den Hauptdiagonalelementen entsprechen.

Es soll gezeigt werden, dass das EDBD-Problem bzgl. V_c , das EDBD-Problem bzgl. V_r (Färbung der Knoten aus V_r und nicht aus V_c) und das ZDBD-Problem jeweils dieselbe minimale Farbanzahl haben, wenn die Farbe 0 nicht mitgezählt wird. Der zusätzliche Buchstabe „D“ in

den Problembezeichnungen steht dafür, dass nur die Hauptdiagonalelemente bestimmt werden sollen.

Nachdem dieser Beweis geführt worden ist, wird nur noch das EDBD-Problem bzgl. V_c betrachtet und gezeigt, dass dieses Problem NP-vollständig ist. Es wird zudem vermutet, dass das ZDBD-Problem NP-vollständig ist. Hierfür wird allerdings kein Beweis angegeben.

Dazu werden zuerst die Färbungen und anschließend die Probleme definiert, wobei die Minimierungsprobleme hier weggelassen werden. Danach wird der Beweis geliefert, dass die minimalen Farbanzahlen der Probleme immer gleich sind (ohne Farbe 0). Zum Schluss wird die NP-Vollständigkeit des EDBD-Problems bzgl. V_c gezeigt.

Problem 3.2.9. (EDBD-Problem bzgl. V_c) Gegeben sei ein bipartiter Graph $G_b = (V_r, V_c, E)$ mit $|V_r| = |V_c|$, p eine Farbanzahl mit $p \geq 3$ und $E_R = \{(r_i, c_i) : i = 1, \dots, |V_c|\}$ die Menge der Kanten, die den benötigten Elementen entsprechen. Existiert eine *einseitige Distanz-2 Färbung* von G_b bzgl. V_c beschränkt auf E_R mit p Farben?

Problem 3.2.10. (EDBD-Problem bzgl. V_r) Gegeben sei ein bipartiter Graph $G_b = (V_r, V_c, E)$ mit $|V_r| = |V_c|$, p eine Farbanzahl mit $p \geq 3$ und $E_R = \{(r_i, c_i) : i = 1, \dots, |V_r|\}$ die Menge der Kanten, die den benötigten Elementen entsprechen. Existiert eine *einseitige Distanz-2 Färbung* von G_b bzgl. V_r beschränkt auf E_R mit p Farben?

Problem 3.2.11. (ZDBD-Problem) Sei $G_b = (V_r, V_c, E)$ ein bipartiter Graph mit $|V_r| = |V_c|$, p eine Farbanzahl mit $p \geq 3$ und $E_R = \{(r_i, c_i) : i = 1, \dots, |V_c|\}$ die Menge der Kanten, die den benötigten Elementen entsprechen. Existiert ein *Star Bicoloring* von G_b beschränkt auf E_R mit p Farben?

Nun wird gezeigt, dass bei dem EDBD-Problem bzgl. V_c , dem EDBD-Problem bzgl. V_r und dem ZDBD-Problem die minimale Farbanzahl p immer identisch ist. Die Beweisidee geht auf [4] zurück.

Lemma 3.2.12. Sei $G_b = (V_r, V_c, E)$ ein bipartiter Graph mit $|V_r| = |V_c|$. Zusätzlich sei die Menge $E_R = \{(c_i, r_i) : i = 1, \dots, |V_r|\}$ gegeben. Die minimale Farbanzahl bei dem EDBD-Problem bzgl. der Spalten, bei dem EDBD-Problem bzgl. der Zeilen und bei dem ZDBD-Problem ist identisch, wenn die Farbe 0 nicht mitgezählt wird.

Beweis. Zuerst wird gezeigt, dass die minimale Farbanzahl bei dem EDBD-Problem bzgl. V_c und bei dem EDBD-Problem bzgl. V_r gleich sind. Danach zeigen wir, dass das gleiche für das EDBD-Problem bzgl. V_c und für das ZDBD-Problem gilt. Die Farbe 0 wird dabei nicht mitgezählt, da sie nicht für einen AD-Berechnungsdurchlauf steht.

Behauptung 1: Die Abbildung $\Phi: V_r \rightarrow \{1, \dots, p\}$ ist genau dann eine *einseitige Distanz-2 Färbung* von G_b auf V_r beschränkt auf E_R , wenn $\Phi': V_c \rightarrow \{1, \dots, p\}$ eine *einseitige Distanz-2 Färbung* von G_b bzgl. V_c beschränkt auf E_R ist, wobei $\Phi(r_i) = \Phi'(c'_i)$, $r_i \in V_r$, $c_i \in V_c$ und $i = 1, \dots, |V_r|$ gilt.

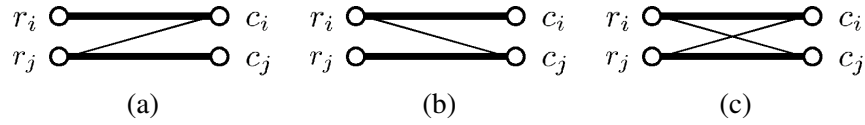


Abbildung 3.2.4: Drei Kombinationen, wie die Knoten c_i und c_j über einen Pfad der Länge zwei verbunden sein können. Die Knoten r_i und r_j sind ebenso über einen Pfad der Länge zwei verbunden

\Leftrightarrow : Es ist zu zeigen, dass ein Knoten $c_i \in V_c$ genau dann einen Distanz-2 Nachbarn $c_j \in V_c$ hat, wenn der Knoten $r_i \in V_r$ einen Distanz-2 Nachbarn $r_j \in V_r$ hat. Die drei möglichen Fälle werden in Abb. 3.2.4 gezeigt. Dadurch, dass die Knoten c_i, r_i und die Knoten c_j, r_j jeweils durch eine Kante, die in E_R enthalten ist, verbunden sind, gibt es einen Pfad (c_i, r_j, c_j) mit $(r_j, c_j) \in E_R$ und/oder einen Pfad (c_i, r_i, c_j) mit $(c_i, r_i) \in E_R$. Somit gilt wegen Bedingung 2 (B2) der Def. 3.2.1 $\Phi'(c_i) \neq \Phi'(c_j)$. Die Knoten r_i und r_j sind dementsprechend auch durch den Pfad (r_i, c_j, r_j) und/oder den Pfad (r_i, c_i, r_j) miteinander verbunden, so dass $\Phi(r_i) \neq \Phi(r_j)$ gilt.

Also gilt für alle Kombinationen von Knoten $c_i, c_j \in V_c$ die Bedingung $\Phi'(c_i) \neq \Phi'(c_j)$ genau dann, wenn $\Phi(r_i) \neq \Phi(r_j)$ gilt. Somit kann weder die Spaltenfärbung noch die Zeilenfärbung nach (B1) aus Def. 3.2.1 weniger Farben als die jeweils andere Färbung benötigen. Das bedeutet, dass es eine Färbung Φ mit minimaler Farbanzahl genau dann gibt, wenn es eine Färbung Φ' mit minimaler Farbanzahl gibt, wobei die beiden Farbanzahlen identisch sind.

Behauptung 2: Gegeben seien ein *Star Bicoloring* von G_b beschränkt auf E_r mit minimaler Farbanzahl und eine *einseitige Distanz-2 Färbung* von G_b bzgl. V_c beschränkt auf E_R mit minimaler Farbanzahl. Dann sind die beiden Farbanzahlen identisch, wobei die Farbe 0 nicht mitgezählt wird.

Gegeben sei ein *Star Bicoloring* $\Phi: [V_r \cup V_c] \rightarrow \{0, 1, \dots, p\}$ von G_b beschränkt auf E_R mit minimaler Farbanzahl. Nun soll gezeigt werden, dass es ein *Star Bicoloring* $\Phi': [V_r \cup V_c] \rightarrow \{0, 1, \dots, p\}$ von G_b beschränkt auf E_R mit minimaler Farbanzahl gibt, bei dem alle Knoten aus V_r mit der Farbe 0 gefärbt sind. Diese Abbildung Φ' ist zugleich eine *einseitige Distanz-2 Färbung* von G'_b bzgl. V_c beschränkt auf E_R mit minimaler Farbanzahl.

Zuerst werden die Kanten $(r_i, c_i) \in E$, $r_i \in V_r$, $c_i \in V_c$ und $i = 1, \dots, |V_r|$, betrachtet, deren Knoten r_i und c_i nur einen Nachbarn haben. Da diese Kanten in E_R enthalten sind, muss bei einer minimalen Färbung genau einer der Knoten mit einer Farbe $\neq 0$ gefärbt sein.

Nun werden die Pfade (r_i, c_i, r_j, c_j) betrachtet, die mit minimaler Farbanzahl gefärbt sind und für deren Knoten entweder die Bedingungen $\Phi(c_i) \neq 0$, $\Phi(c_i) = \Phi(c_j)$ und $\Phi(r_j) \neq 0$ oder die Bed. $\Phi(r_i) \neq 0$, $\Phi(r_i) = \Phi(r_j)$ und $\Phi(c_i) \neq 0$ gelten (z.B. die Pfade (e) und (f) in Abb. 3.2.5). Diese Pfade können so gefärbt werden, dass im 1. Fall $\Phi(c_j) = 0$ oder im 2. Fall $\Phi(r_j) = 0$ gilt und die Färbung immer noch ein *beschränktes Star Bicoloring* mit minimaler Farbanzahl ist. Die Pfade mit den resultierenden Färbungen sind die Pfade (c) und (f).

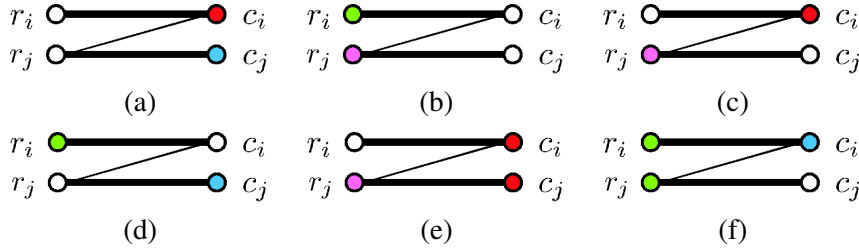


Abbildung 3.2.5: Alle sechs Möglichkeiten (ohne Farbpermutationen), einen Pfad der Länge drei mit der minimalen Farbanzahl einzufärben

Somit ist von jeder Kante aus E_R nur noch ein Knoten mit einer Farbe $\neq 0$ gefärbt. Nun können die Knoten c_i , $i = 1, \dots, |V_c|$, mit den Farben der Knoten r_i , $i = 1, \dots, |V_r|$, gefärbt werden, und den Knoten r_i wird die Farbe 0 zugewiesen. Dadurch werden Bedingung 1 (B1) und (B2) aus Def. 3.2.5 nicht verletzt, da nun alle Knoten aus V_r mit der Farbe 0 gefärbt sind und immer noch von jeder Kante aus E_R ein Knoten mit einer Farbe $\neq 0$ gefärbt ist. Als Beispiel betrachten wir die Pfade (c), (d) und (a) in Abb. 3.2.5. Das bedeutet, dass jedes *beschränkte Star Bicoloring* Φ mit minimaler Farbanzahl, bei dem beide Knotenmengen mit Farben $\neq 0$ gefärbt sind, in ein minimales, *beschränktes Star Bicoloring* Φ' , bei dem nur die Knotenmenge V_c mit Farben $\neq 0$ gefärbt ist, überführt werden kann.

Wenn alle Knoten in V_r mit der Farbe 0 gefärbt sind, dann muss nach (B3) aus Def. 3.2.5 für alle Distanz-2 Nachbarn $c_i, c_j \in V_c$ die Bedingung $\Phi(c_i) \neq \Phi(c_j)$ gelten. Die Abbildung Φ' ist somit auch eine *einseitige Distanz-2 Färbung* von G'_b bzgl. V_c beschränkt auf E_R mit minimaler Farbanzahl. Hieraus kann direkt abgeleitet werden, dass jede *einseitige Distanz-2 Färbung* von G_b bzgl. V_c beschränkt auf E_R ein *Star Bicoloring* von G_b ist.

Somit wurde die Behauptung 2 bewiesen. Mit dem Beweis von Behauptung 1 und 2 wurde damit auch das Lemma gezeigt. \square

Durch eine Reduktion vom p -Färbbarkeitsproblem 2.4.4 auf das EDBD-Problem wird gezeigt, dass das EDBD-Problem bzgl. V_c NP-vollständig ist. Der Beweis für das EDBD-Problem bzgl. V_r wäre genauso zu führen.

Lemma 3.2.13. *Das EDBD-Problem bzgl. V_c (Problem 3.2.9) ist NP-vollständig.*

Beweis. Zuerst muss gezeigt werden, dass das EDBD-Problem \in NP ist: Gegeben sei eine Eingabe (G'_b, E'_R, p') , $G'_b = (V'_r, V'_c, E')$, des EDBD-Problems. Es wird eine *einseitige Distanz-2 Färbung* von G'_b bzgl. V'_c beschränkt auf E'_R mit $E'_R = \{(r'_i, c'_i) : i = 1, \dots, |V'_c|\}$ geraten. Dann muss in polynomieller Zeit überprüft werden, ob die Knotenfärbung eine Bedingung der Def. 3.2.1 verletzt.

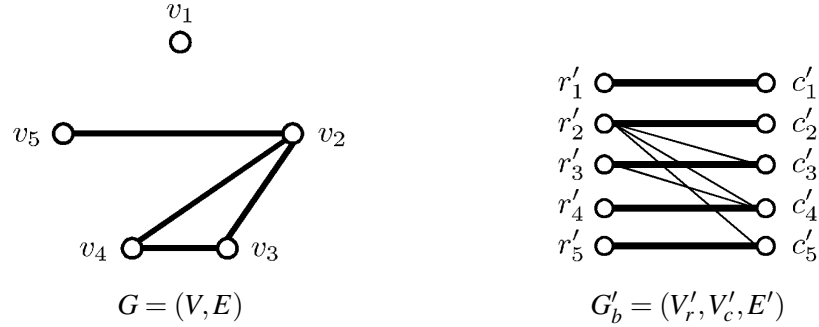


Abbildung 3.2.6: Transformation eines Graphen G der Eingabe des p -Färbbarkeitsproblems 2.4.4 in einen Graphen G'_b der Eingabe des EDBD-Problems 3.2.9

Dazu kann der polynomielle Algorithmus verwendet werden, der diese Überprüfung in Lemma 3.2.4 vornimmt. Zusätzlich muss noch in $O(|E'|)$ überprüft werden, ob alle Kanten $(r'_i, c'_i) \in E'$, $r'_i \in V'_r$, $c'_i \in V'_c$ und $i = 1, \dots, |V'|$, auch in E'_R enthalten sind. Der Gesamtaufwand ist damit $O(|E'| \cdot \Delta_1(V'_r))$ und polynomiell.

Nun betrachten wir die polynomielle Reduktion. Sei $G = (V, E)$ und p eine Eingabe des p -Färbbarkeitsproblems (Problem 2.4.4). Diese Eingabe wird in eine Eingabe (G'_b, E'_R, p') , $G'_b = (V'_r, V'_c, E')$, des Problems 3.2.9 transformiert (vgl. Beispiel in Abb. 3.2.6):

- (T1) $V'_r = \{r'_i : i = 1, \dots, |V|\}$, $V'_c = \{c'_i : i = 1, \dots, |V|\}$,
- (T2) $E' = \{(r'_i, c'_j) : \forall (v_i, v_j) \in E, i, j = 1, \dots, |V|, i < j\} \cup \{(r'_i, c'_i) : i = 1, \dots, |V|\}$,
- (T3) $E'_R = \{(r'_i, c'_i) : i = 1, \dots, |V|\}$,
- (T4) $p' = p + 1$.

Die Transformation ist in polynomieller Zeit möglich: Jeder Knoten $v_i \in V$, $i = 1, \dots, |V|$, wird besucht und dabei werden alle Kanten zu noch nicht besuchten Nachbarknoten betrachtet. Für jeden Knoten v_i werden zwei Knoten und eine Kante zum Graphen G'_b und für jeden entsprechenden Nachbarknoten wird eine Kante zum Graphen G'_b hinzugefügt. Daraus folgt eine Abschätzung von $O(|V| \cdot \frac{1}{2} \bar{\delta}_1(V)) = O(|E|)$.

Es ist zu zeigen, dass $(G, p) \in p$ -Färbbarkeitsproblem $\Leftrightarrow (G'_b, E'_R, p') \in$ Problem 3.2.9 ist; präziser: die Abbildung $\Phi : V \rightarrow \{1, \dots, p\}$ ist genau dann eine *Distanz-1 Färbung* von G , wenn $\Phi' : V'_c \rightarrow \{0, \dots, p' - 1\}$ eine *einseitige Distanz-2 Färbung* von G'_b bzgl. V'_c beschränkt auf E'_R ist, wobei für die Knoten $v_i \in V$ und $c'_i \in V'_c$, $i = 1, \dots, |V|$, die Bedingung $\Phi(v_i) = \Phi'(c'_i)$ gilt.

\Leftrightarrow : Die Knoten $v_i, v_j \in V$, $i, j = 1, \dots, |V|$, sind genau dann durch eine Kante $(v_i, v_j) \in E$ verbunden, wenn die entsprechenden Knoten $c'_i, c'_j \in V'_c$ nach (T2) über den Pfad (c'_i, r'_i, c'_j) , $r'_i \in V'_r$, miteinander verbunden sind, wobei die Kante $(c'_i, r'_i) \in E'_R$ ist. Somit muss wegen (B1) aus Def. 3.1.1 und (B2) aus Def. 3.2.1 gelten: $\Phi(v_i) \neq \Phi(v_j) \Leftrightarrow \Phi'(c'_i) \neq \Phi'(c'_j)$. Vgl. beispielsweise Knoten $v_2, v_4 \in V$ und $c'_2, c'_4 \in V'_c$ in Abbildung 3.2.6.

Keinem Knoten $c'_i \in V'_c$, $i = 1, \dots, |V'_c|$, darf wegen (B1) aus Def. 3.2.1 die Farbe 0 zugewiesen werden, da dieser Knoten nach (T3) inzident zu einer Kante aus E'_R ist. Die Bedingung $\Phi(v_i) = \Phi'(c'_i)$ kann somit nie dadurch verletzt werden, dass dem Knoten c'_i die Farbe 0 zugewiesen wird.

Somit müssen zwei Knoten aus V genau dann unterschiedlich gefärbt werden, wenn die entsprechenden zwei Knoten in V'_c unterschiedlich gefärbt werden müssen. Weil zudem kein Knoten aus V'_c mit der Farbe 0 gefärbt werden darf, ist die Abbildung Φ genau dann eine *Distanz-1 Färbung* von G , wenn Φ' eine *einseitige Distanz-2 Färbung* von G'_b bzgl. V'_c beschränkt auf E'_R ist, wobei die Bedingung $\Phi(v_i) = \Phi'(c'_i)$, $i = 1, \dots, |V|$, gilt. \square

Damit ist gezeigt, dass das Entscheidungsproblem 3.2.9 NP-vollständig ist.

Es wurde in Lemma 3.2.12 gezeigt, dass ein *Star Bicoloring* von G_b mit minimaler Farbanzahl in eine *einseitige Distanz-2 Färbung* von G_b bzgl. V_c mit minimaler Farbanzahl überführt werden kann. Wenn es für diese Überführung einen polynomiellen Algorithmus gibt (wovon auszugehen ist), dann muss auch das ZDBD-Problem NP-vollständig sein, da ansonsten das EDBD-Problem nicht mehr NP-vollständig wäre.

Somit kann davon ausgegangen werden, dass es keinen polynomiellen Algorithmus zum Färben eines bipartiten Graphen $G_b = (V_r, V_c, E)$ gibt, wenn die Menge E_R die Kanten enthält, die den Hauptdiagonalelementen entsprechen.

4 Färbungsheuristiken

Zur unidirektionalen, direkten und vollständigen Bestimmung bzw. zur bidirektionalen, direkten und vollständigen Bestimmung einer Jacobi-Matrix J werden zwei Heuristiken vorgestellt, die den entsprechenden bipartiten Graphen $G_b = (V_r, V_c, E)$ so färben, dass das Resultat eine *einseitige Distanz-2 Färbung von G_b bzgl. V_c* bzw. ein *Star Bicoloring von G_b* ist. Diese Heuristiken liefern im Allgemeinen keine Färbungen mit minimaler Farbanzahl zurück. Im Folgenden wird bei so einer Färbung auch von minimaler Färbung gesprochen.

Zur partiellen Bestimmung der Jacobi-Matrix J sei zusätzlich die Kantenmenge E_R gegeben, wobei die enthaltenen Kanten den benötigten Elementen von J entsprechen. Die beiden Heuristiken zur vollständigen Bestimmung werden in der vorliegenden Arbeit so angepasst, dass sie eine *einseitige Distanz-2 Färbung von G_b bzgl. V_c beschränkt auf E_R* bzw. ein *Star Bicoloring von G_b beschränkt auf E_R* berechnen.

Aus den resultierenden Färbungen können jeweils eine Seed-Matrix S oder zwei Seed-Matrizen S_r und S_c bestimmt werden, so dass die (benötigten) Elemente einer Jacobi-Matrix J direkt aus der komprimierten Jacobi-Matrix $\tilde{J} = J \cdot S$ bzw. $\tilde{J} = S_r \cdot J \cdot S_c$ extrahiert werden können.

Sowohl für die einseitigen als auch für die zweiseitigen Färbungen werden zudem Heuristiken betrachtet, die die Knoten des Graphen G_b vor dem Färben sortieren. Dadurch soll die Anzahl der Farben, die die Färbungsheuristiken benötigen, reduziert werden.

Die Färbungsheuristiken und Vorsortierungsalgorithmen wurden für die vorliegende Arbeit in der Programmiersprache C++ unter Zuhilfenahme der *Boost Graph Library* [23] implementiert. Mit diesen Implementierungen wurden auch die Ergebnisse des nächsten Kapitels berechnet. In diesem Kapitel werden die Algorithmen allerdings im sogenannten Pseudocode angegeben.

4.1 Einseitige Färbungen

Zuerst wird eine einseitige Färbungsheuristik für bipartite Graphen $G_b = (V_r, V_c, E)$ vorgestellt, die eine *einseitige Distanz-2 Färbung von G_b bzgl. V_c* bestimmt. Anschließend wird diese Heuristik so abgewandelt, dass sie eine *einseitige Distanz-2 Färbung von G_b bzgl. V_c beschränkt auf E_R* zurückliefert.

Am Ende dieses Abschnitts werden drei Vorsortierungsalgorithmen für die vollständige Fär-

bung vorgestellt. Da sie im Kontext des „column intersection graph“ (CIG) $G_c = (V, E)$ entwickelt worden sind, müssen sie für bipartite Graphen $G_b = (V_r, V_c, E)$ angepasst werden. Zudem müssen diese Algorithmen auf den beschränkten Fall angepasst werden.

4.1.1 Vollständige Färbung

Die Färbungsheuristik *EinseitigeD2Färbung* (Alg. 4.1.1) wurde in [13, Alg. 3.2] als Algorithmus *PartialD2Coloring* vorgestellt und berechnet eine *einseitige Distanz-2 Färbung* von G_b bzgl. V_c (Def. 2.4.6). Aus einer solchen Färbung kann eine Seed-Matrix für den Vorwärtsmodus von AD bestimmt werden. Der Algorithmus könnte ebenso die Knoten aus V_r färben, wobei dieser Fall hier nicht betrachtet wird.

Diese Heuristik, die einen bipartiten Graphen $G_b = (V_r, V_c, E)$ färbt, ist eine Adaption der Färbungsheuristik *sequential coloring algorithm* (SEQ) aus [6], die eine *Distanz-1 Färbung* von G_c (CIG) berechnet. Der Färbungsalgorithmus für CIGs ist eine Graphmodellierung des Partitionierungsalgorithmus der CPR-Technik (Abschnitt 2.4).

Der Algorithmus färbt die Knoten aus V_c mit Farben aus der Menge $\{1, \dots, |V_c|\}$. Nacheinander werden die Knoten $c_i \in V_c$, $i = 1, \dots, |V_c|$, besucht, und sie erhalten jeweils die kleinste Farbe, mit der kein Distanz-2 Nachbar von c_i gefärbt ist.

Die Heuristik in Alg. 4.1.1 ist folgendermaßen implementiert: Die Farbe eines Knotens $c_i \in V_c$, $i \in \{1, \dots, |V_c|\}$, wird im Array *Farbe* an der Stelle i gespeichert. Über alle Knoten $c_i \in V_c$ wird in der gegebenen Reihenfolge $i = 1, \dots, |V_c|$ iteriert. Von jedem Knoten c_i werden alle gefärbten Distanz-2 Nachbarn x betrachtet, wobei die Funktion $N_2(c_i, G_b)$ in Zeile 5 alle Distanz-2 Nachbarn des Knotens c_i zurückliefert. Die Farben der Nachbarn x werden in Zeile 6 im Array *verboteneFarben* gesperrt, welches für jede Farbe der Menge $\{1, \dots, |V_c|\}$ ein Element hat. Dem Element *verboteneFarben*[*Farbe*[x]] wird der Index i zugewiesen, um die Farbe des Knotens x für den Knoten c_i zu sperren. Deswegen musste in Zeile 3 jedes Element dieses Arrays mit einem Wert initialisiert werden, der keinem Knotenindex entspricht. Nach dem Betrachten aller Distanz-2 Nachbarn x erhält der Knoten c_i in Zeile 8 den Index des niedrigsten Elements in *verboteneFarben* als Farbe zugewiesen, dessen Wert nicht i ist.

In [13] wird gezeigt, dass die resultierende Färbung eine *einseitige Distanz-2 Färbung* von G_b bzgl. V_c ist. Die Laufzeit wird mit $O(|V_c| \cdot \bar{\delta}_1(V_c) \cdot \Delta_1(V_r)) = O(|V_c| \cdot \bar{\delta}_2(V_c))$ abgeschätzt, wobei $\bar{\delta}_k(V)$ den durchschnittlichen Knotengrad der Knoten aus V bzgl. der Distanz- k Nachbarn und $\Delta_k(V)$ den entsprechenden maximalen Knotengrad bezeichnet.

Diese Abschätzung ist offensichtlich korrekt, da von allen Knoten $v \in V_c$ die Distanz-1 Nachbarn aus V_r besucht werden. Von diesen wiederum werden die Distanz-1 Nachbarn aus V_c (also die Distanz-2 Nachbarn des Ursprungsknotens v) betrachtet.

In [24] wird die obere Schranke

$$\max\{\min[d_1(v_i) + 1, i] : v_i \in V, 1 \leq i \leq |V|\}$$

Algorithmus 4.1.1 Heuristik zur einseitigen Distanz-2 Färbung von G_b bzgl. V_c

```

1: procedure EINSEITIGED2FÄRBUNG( $G_b = (V_r, V_c, E)$ )
2:   Sei  $c_1, c_2, \dots, c_{|V_c|}$  eine gegebene Reihenfolge von  $V_c$ 
3:   Initialisiere Array Farbe mit  $-1$  und Array verboteneFarben mit  $-1$ 
4:   for  $i := 1, \dots, |V_c|$  do
5:     for each  $x \in N_2(c_i, G_b)$  mit  $Farbe[x] > 0$  do
6:        $verboteneFarben[Farbe[x]] := i$ 
7:     end for
8:      $Farbe[c_i] := \min\{j > 0: verboteneFarben[j] \neq i\}$ 
9:   end for
10:  return Farbe
11: end procedure

```

für die maximale Farbanzahl der Färbung des CIG $G_c = (V, E)$ angegeben, wobei $d_k(v)$ den Knotengrad bzgl. der Distanz- k Nachbarn von v bezeichnet. Übertragen auf den bipartiten Graphen $G_b = (V_r, V_c, E)$ lautet die Abschätzung dementsprechend:

$$\max\{\min[d_2(c_i) + 1, i] : c_i \in V_c, 1 \leq i \leq |V_c|\}.$$

Diese Schranke kann direkt anhand der Färbungsheuristik nachvollzogen werden: Einem Knoten $c_i \in V_c$ wird im i -ten Schritt höchstens die Farbe i zugewiesen, da die vorherigen Knoten höchstens mit den Farben $1, \dots, i-1$ gefärbt sein können. Wenn der Knoten c_i allerdings den Knotengrad $d_2(c_i)$ hat und die Bedingung $d_2(c_i) + 1 < i$ gilt, dann werden höchstens $d_2(c_i) + 1$ Farben benötigt. Die benötigte Farbanzahl ist genau $d_2(c_i) + 1$, wenn alle Distanz-2 Nachbarn unterschiedlich gefärbt sind.

Somit ist das Minimum von $d_2(c_i) + 1$ und dem Index i eine obere Schranke bzgl. der benötigten Farben für den Knoten c_i im Schritt i . Das Maximum aller dieser oberen Schranken gibt die höchste Anzahl von Farben an, die die Graphfärbungsheuristik benötigt.

4.1.2 Beschränkte Färbung

Die Heuristik *BeschränkteEinseitigeD2Färbung* (Alg. 4.1.2) ist eine Anpassung der Heuristik *EinseitigeD2Färbung* (Alg. 4.1.1), so dass nicht mehr alle Elemente der Jacobi-Matrix J direkt bestimmt werden müssen, sondern dass bei der partiellen Berechnung nur noch alle benötigten Elemente direkt bestimmt werden müssen. Dazu wird die Färbungsheuristik auf die Menge E_R beschränkt, damit dieser Algorithmus eine *einseitige Distanz-2 Färbung von G_b bzgl. V_c beschränkt auf E_R* (Def. 3.2.1) berechnet.

Diese Heuristik färbt nur diejenigen Distanz-2 Nachbarn unterschiedlich, die durch einen Pfad der Länge zwei miteinander verbunden sind, von dem mindestens eine Kante in der Menge

E_R enthalten ist. Diese Nachbarn werden hier als *Distanz-2 Nachbarn beschränkt durch E_R* oder kurz als *beschränkte Distanz-2 Nachbarn* bezeichnet.

Algorithmus 4.1.2 Heuristik zur einseitigen Distanz-2 Färbung von G_b bzgl. V_c beschränkt auf E_R

```

1: procedure BESCHRÄNKTEINSEITIGED2FÄRBUNG( $G_b = (V_r, V_c, E), E_R$ )
2:   Sei  $c_1, c_2, \dots, c_{|V_c|}$  eine gegebene Reihenfolge von  $V_c$ 
3:   Initialisiere Array Farbe mit  $-1$  und Array verboteneFarben mit  $-1$ 
4:   for  $i := 1, \dots, |V_c|$  do
5:     for each  $x \in N_2\text{beschränkt}(c_i, G_b, E_r)$  mit  $\text{Farbe}[x] > 0$  do
6:        $\text{verboteneFarben}[\text{Farbe}[x]] := i$ 
7:     end for
8:      $\text{Farbe}[c_i] := \min\{j > 0 : \text{verboteneFarben}[j] \neq i\}$ 
9:   end for
10:  return Farbe
11: end procedure

```

Der Algorithmus 4.1.2 unterscheidet sich nur in dem Punkt von der Heuristik 4.1.1, dass nicht mehr alle Distanz-2 Nachbarn eines Knotens betrachtet werden, sondern nur noch die beschränkten Distanz-2 Nachbarn. Dazu wird in Zeile 5 die Funktion $N_2(v, G_b)$ in Alg. 4.1.1 durch die Funktion $N_2\text{beschränkt}(v, G_b, E_r)$, die die beschränkten Distanz-2 Nachbarn von v zurückliefert, ersetzt.

Lemma 4.1.1. *Die Heuristik BeschränkteEinseitigeD2Färbung berechnet eine einseitige Distanz-2 Färbung von G_b bzgl. V_c beschränkt auf E_R in $O(|V_c| \cdot \tilde{\delta}_{2,E_R}(V_c))$, wobei $\tilde{\delta}_{2,E_R}(V_c)$ den durchschnittlichen Knotengrad der Knoten aus V_c bzgl. der beschränkten Distanz-2 Nachbarn bezeichnet.*

Beweis. Korrektheit: Jeder Knoten $c_i \in V_c$, $i = 1, \dots, |V_c|$, wird mit einer Farbe $\neq 0$ gefärbt. Somit gilt für den Knoten c_i jeder Kante $(c_i, r_j) \in E_R$ die Bedingung $\Phi(c_i) \neq 0$.

Des Weiteren muss für alle Pfade (c_i, w, x) mit $c_i, x \in V_c$, $w \in V_r$ und $(c_i, w) \in E_R$ oder $(w, x) \in E_R$ die Bedingung $\Phi(c_i) \neq \Phi(x)$ gelten. Im Array *verboteneFarben* werden die Farben der beschränkten Distanz-2 Nachbarn x des Knotens c_i für diesen gesperrt. Anschließend wird dem Knoten c_i die kleinste nicht gesperrte Farbe zugewiesen.

Abschätzung: In jedem Schritt werden ein Knoten $c_i \in V_c$ und alle seine beschränkten Distanz-2 Nachbarn $c_j \in V_c$, die schon gefärbt sind, betrachtet. Dieses führt direkt zu einer Abschätzung der Laufzeit von $O(|V_c| \cdot \tilde{\delta}_{2,E_R}(V_c))$. \square

Die obere Schranke

$$\max\{\min[d_{2,E_R}(c_i) + 1, i] : c_i \in V_c, 1 \leq i \leq |V_c|\}$$

für die Farbanzahl geht direkt auf die obere Schranke von Algorithmus 4.1.1 zurück. Die Schranke ist somit auf die beschränkten Distanz-2 Nachbarn angepasst worden.

4.1.3 Vorsortierung

Die Knoten der zu färbenden Knotenmenge sollen nun durch verschiedene Algorithmen vorsortiert werden, um die Anzahl der Farben, die von den Färbungsheuristiken benötigt werden, möglicherweise zu reduzieren. Dazu werden drei unterschiedliche Algorithmen betrachtet.

Bei diesen Algorithmen handelt es sich um das *Largest-First Ordering (LFO)*, das *Smallest-Last Ordering (SLO)* und das *Incidence Degree Ordering (IDO)*. Diese Algorithmen werden zunächst als Vorsortierung für die Heuristik *EinseitigeD2Färbung* (Alg. 4.1.1) eingeführt. Anschließend werden sie für die Heuristik *BeschränkteEinseitigeD2Färbung* (Alg. 4.1.2) angepasst.

Diese Vorsortierungsalgorithmen wurden ursprünglich im Kontext der SEQ-Heuristik zur Färbung von CIG $G_c = (V, E)$ entwickelt und müssen nun auf die bipartiten Graphen $G_b = (V_r, V_c, E)$ angepasst werden. Die Korrektheit der zwei Färbungsheuristiken wird durch die Vorsortierung nicht verletzt, da die Knotenreihenfolgen beliebig sein dürfen.

Sei nun die Knotenmenge V_c eines bipartiten Graphen $G_b = (V_r, V_c, E)$ gegeben, wobei die Reihenfolge der Knoten beliebig ist. In Abschnitt 4.1.1 wurde für die Anzahl der Farben, die die Heuristik *EinseitigeD2Färbung* benötigt, eine obere Schranke von

$$\max\{\min[d_2(c_i) + 1, i] : c_i \in V_c, 1 \leq i \leq |V_c|\}$$

angegeben. Bei dem Algorithmus *Largest-First Ordering (LFO)* in Alg. 4.1.3, der auf [24] zurückgeht, werden die Knoten so sortiert, dass sich die Farbanzahl unter Berücksichtigung der oberen Schranke oftmals verringert. Der Algorithmus wird vom CIG $G_c = (V, E)$ auf den bipartiten Graphen $G_b = (V_c, V_r, E)$ angepasst, indem nicht mehr die Knoten der Knotenmenge V , sondern die Knoten von V_c sortiert werden. Zusätzlich werden entsprechend keine Distanz-1 Nachbarn, sondern Distanz-2 Nachbarn betrachtet. Diese Anpassung gilt auch für die beiden anderen Vorsortierungsalgorithmen.

Der *LFO*-Algorithmus sortiert die Knoten $c_i \in V_c$, $i = 1, \dots, |V_c|$, nicht aufsteigend nach Anzahl der Distanz-2 Nachbarn. Je höher der Knotengrad $d_2(c_i)$ ist, desto eher erscheint der Knoten c_i in der sortierten Reihenfolge. Sollten zwei oder mehr Knoten denselben Knotengrad haben, dann wird der Knoten $c_i \in V_c$ mit dem größten Index i ausgewählt.

Bei der Sortierung wird ausgenutzt, dass in der Schranke für jeden Knoten $c_i \in V_c$, $i = 1, \dots, |V_c|$, das Minimum von $d_2(c_i) + 1$ und dem Index i genommen wird. Durch die Sortierung des *LFO*-Algorithmus wird versucht, die Minima insgesamt zu verringern. Dazu werden die Knoten mit hohem Knotengrad vorne in der Reihenfolge eingefügt, damit sie einen niedrigen Index i erhalten, und somit der Index als Minimum zurückgeliefert wird und nicht $d_2(c_i) + 1$.

Algorithmus 4.1.3 Largest-First Ordering

```

1: procedure LFO( $G_b = (V_r, V_c, E)$ )
2:   Sei  $c_1, c_2, \dots, c_{|V_c|}$  eine beliebige Reihenfolge der Knotenmenge  $V_c$ 
3:   for  $i := 1, \dots, |V_c|$  do
4:      $Reihenfolge[i] := v$ , wobei  $v \in V_c$  höchsten Grad  $d_2(v) := |N_2(v, G_b)|$  hat
5:     Entferne den Knoten  $v$  aus  $V_c$ 
6:   end for
7:   return  $Reihenfolge$ 
8: end procedure

```

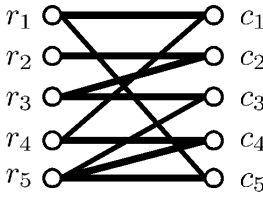
	Algorithmus	Reihenfolge, Färbung $[\Phi(c_1), \dots] = [1, \dots]$	#Farben
 $G_b = (V_r, V_c, E)$	–	$c_1, c_2, c_3, c_4, c_5,$ $[c_1, c_2, c_3, c_4, c_5] = [1, 1, 2, 3, 4]$	4
	<i>LFO</i>	$c_5, c_4, c_3, c_1, c_2,$ $[c_1, c_2, c_3, c_4, c_5] = [1, 3, 3, 2, 1]$	3
	<i>SLO</i>	$c_5, c_4, c_3, c_1, c_2,$ $[c_1, c_2, c_3, c_4, c_5] = [1, 3, 3, 2, 1]$	3
	<i>IDO</i>	$c_3, c_4, c_5, c_1, c_2,$ $[c_1, c_2, c_3, c_4, c_5] = [1, 2, 1, 2, 3]$	3

Abbildung 4.1.1: Färbungen der Knoten von V_c des bipartiten Graphen $G_b = (V_c, V_r, E)$ mit der Heuristik *EinseitigeD2Färbung* in unterschiedlichen Reihenfolgen (nicht vorsortiert, mit *LFO*, *SLO* und *IDO* vorsortiert)

Knoten mit niedrigem Knotengrad kommen weiter nach hinten in die Reihenfolge, da hier der Index i oftmals größer ist als $d_2(c_i) + 1$.

In Abb. 4.1.1 ist ein Vergleich zwischen den verschiedenen Vorsortierungsheuristiken zu sehen. Bei diesem kleinen Beispiel wird durch die Vorsortierung mit dem *LFO*-Algorithmus eine Farbe bei der anschließenden Färbung gespart.

Der zweite Algorithmus wird in [19] als *Smallest-Last Ordering* (*SLO*) (Alg. 4.1.4) vorgestellt. Dieser Algorithmus baut die Knotenreihenfolge rückwärts auf. Dazu wird in jedem Schritt derjenige Knoten aus V_c am Anfang der sortierten Reihenfolge eingefügt, der die wenigsten Distanz-2 Nachbarn hat, die noch nicht zur Reihenfolge hinzugefügt worden sind.

Durch dieses Verfahren ergibt sich folgende obere Abschätzung von

$$\max\{1 + \delta_1(V \setminus \{v_{i+1}, \dots, v_{|V|}\}) : 1 \leq i \leq |V|\}$$

für die SEQ-Heuristik, wobei $\delta_1(G_0)$ den minimalen Knotengrad im Teilgraphen G_0 bezeichnet.

Algorithmus 4.1.4 Smallest-Last Ordering

```

1: procedure SLO( $G_b = (V_c, V_r, E)$ )
2:   Sei  $c_1, c_2, \dots, c_{|V_c|}$  eine beliebige Reihenfolge der Knotenmenge  $V_c$ 
3:   for each  $i := |V_c|, \dots, 1$  do
4:      $Reihenfolge[i] := v$ , wobei  $v \in V_c$  niedrigsten Grad  $d_2(v) := |N_2(v, G_b)|$  hat
5:     Entferne den Knoten  $v$  aus  $V_c$ 
6:   end for
7:   return  $Reihenfolge$ 
8: end procedure

```

Angepasst auf den bipartiten Graphen ergibt sich die Schranke

$$\max\{1 + \delta_2(V_c \setminus \{c_{i+1}, \dots, c_{|V_c|}\}) : 1 \leq i \leq |V_c|\}.$$

Wenn das Beispiel in Abb. 4.1.1 mit dem *SLO*-Algorithmus vorsortiert wird, dann wird eine Farbe eingespart. Im Allgemeinen gilt aber nicht, dass die Algorithmen *LFO*, *SLO* und *IDO* die Farbanzahl genau gleich stark reduzieren.

Der letzte Vorsortierungsalgorithmus (Alg. 4.1.5) ist das *Incidence Degree Ordering (IDO)* aus [3]. Dieser baut die Knotenreihenfolge wieder von vorne auf. Nun werden nacheinander diejenigen Knoten gewählt, von denen die meisten Distanz-2 Nachbarn schon ausgewählt und zur sortierten Reihenfolge hinzugefügt wurden. Dieses Vorgehen spiegelt sich in der Namensgebung wieder. In Abb. 4.1.1 ist zu sehen, dass auch durch die Reihenfolge des *IDO*-Algorithmus eine Farbe gespart wird.

Algorithmus 4.1.5 Incidence Degree Ordering

```

1: procedure IDO( $G_b = (V_r, V_c, E)$ )
2:   Sei  $c_1, c_2, \dots, c_{|V_c|}$  eine beliebige Reihenfolge der Knotenmenge  $V_c$ 
3:   Initialisiere Array  $inzidenterGrad$  mit 0
4:   for  $i := 1, \dots, |V_c|$  do
5:      $Reihenfolge[i] := v$ , wobei  $v \in V_c$  mit höchstem Wert  $inzidenterGrad[v]$ 
6:     for each  $x \in N_2(v, G_b)$  mit  $v \notin Reihenfolge$  do
7:       Inkrementiere  $inzidenterGrad[x]$ 
8:     end for
9:   end for
10:  return  $Reihenfolge$ 
11: end procedure

```

Nun sei ein bipartiter Graph $G_b = (V_r, V_c, E)$ und die Menge E_R gegeben. Dieser Graph soll mit der Heuristik *BeschränkteEinseitigeD2Färbung* gefärbt werden. Damit diese Heuristik bessere Werte liefert, ist es wiederum sinnvoll, die Knotenmenge V_c vorher zu sortieren.

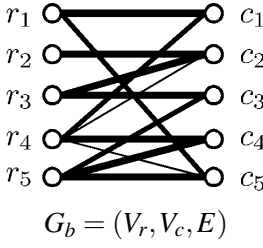
	Algorithmus	Reihenfolge, Färbung $[\Phi(c_1), \dots] = [1, \dots]$	#Farben
 $G_b = (V_r, V_c, E)$	–	$c_1, c_2, c_3, c_4, c_5,$ $[c_1, c_2, c_3, c_4, c_5] = [1, 1, 2, 3, 4]$	4
	<i>LFO</i>	$c_5, c_4, c_2, c_3, c_1,$ $[c_1, c_2, c_3, c_4, c_5] = [4, 3, 4, 2, 1]$	4
	<i>LFObeschränkt</i>	$c_4, c_5, c_3, c_2, c_1,$ $[c_1, c_2, c_3, c_4, c_5] = [1, 2, 1, 2, 3]$	3

Abbildung 4.1.2: Färbungen der Knotenmenge V_c des bipartiten Graphen $G_b = (V_r, V_c, E)$ beschränkt auf E_R mit der Heuristik *BeschränkteEinseitigeD2Färbung* in unterschiedlichen Knotenreihenfolgen (nicht vorsortiert, mit *LFO* und *LFObeschränkt* vorsortiert)

Die Färbungsheuristik *EinseitigeD2Färbung* (Alg. 4.1.1) wurde auf die Färbungsheuristik *BeschränkteEinseitigeD2Färbung* (Alg. 4.1.2) angepasst, indem nicht mehr die Distanz-2 Nachbarn, sondern die beschränkten Distanz-2 Nachbarn betrachtet werden. Dieselbe Anpassung wird bei den Vorsortierungsalgorithmen angewendet, so dass der Knotengrad nicht mehr bzgl. der Distanz-2 Nachbarn, sondern bzgl. der beschränkten Distanz-2 Nachbarn berechnet wird.

Bei den Vorsortierungsalgorithmen muss dazu nur die Funktion $N_2(v, G_b)$ durch die Funktion $N_{2beschränkt}(v, G_b, E_R)$ ersetzt werden. Diese liefert die beschränkten Distanz-2 Nachbarn des Knotens v zurück, so dass der Knotengrad bzgl. dieser Nachbarn in den Algorithmen berechnet wird. Die angepassten Algorithmen werden *LFObeschränkt*, *SLObeschränkt* und *IDObeschränkt* genannt.

In Abb. 4.1.2 ist ein Beispielgraph $G_b = (V_r, V_c, E)$ mit der Beschränkung E_R zu sehen. Durch eine Vorsortierung mit *LFO* verringert sich die Anzahl der benötigten Farben nicht, mit dem Algorithmus *LFObeschränkt* dagegen schon.

4.2 Zweiseitige Färbungen

4.2.1 Vollständige zweiseitige Färbung

Die Heuristik *StarBicoloringSchema* (Alg. 4.2.1) wird in [13, Alg. 5.1] vorgestellt. Dieser Algorithmus färbt beide Knotenmengen des bipartiten Graphen $G_b = (V_r, V_c, E)$ so, dass die berechnete Färbung ein *Star Bicoloring* von G_b (Def. 3.1.4) ist. Auch dieser Algorithmus liefert nicht notwendigerweise eine Färbung mit der minimalen Farbanzahl zurück.

Die Heuristik besteht aus zwei Teilen: Zuerst wird ein Independent Set I (IS I) des Graphen G_b mit dem Algorithmus *ISet* (Alg. 4.2.2) bestimmt. Die Knoten des IS I werden mit der

Farbe 0 gefärbt. Anschließend werden die Knoten des Vertex Cover $C = (V_r \cup V_c) \setminus I$ mit der Heuristik *StarBicoloring* (Alg. 4.2.3) gefärbt.

Algorithmus 4.2.1 Heuristik zur Berechnung eines Star Bicoloring von G_b

procedure STARBICOLORINGSCHEMA($G_b = (V_r, V_c, E)$)

1. Berechnung eines IS I
2. Knoten aus dem IS I werden mit Farbe 0 gefärbt
3. Färben der Knoten aus dem VC $C = (V_r \cup V_c) \setminus I$

end procedure

Bei der Berechnung des IS I ist nicht die Maximierung das Ziel, sondern es ist wichtig das IS I so zu wählen, dass die Knoten im Vertex Cover C (VC C) mit möglichst wenig Farben gefärbt werden können.

Zur Berechnung des IS I wird nicht der in [13, Abschnitt 5.4] beschriebene Algorithmus genutzt, der eine abstrakte graphentheoretische Beschreibung des Algorithmus *MNCO* aus [7] ist, wobei eine bestimmte Auswahlfunktion nicht näher spezifiziert wird. Zur Berechnung des IS I wird ein Algorithmus (Alg. 4.2.2) aus dem Softwaretool ADMIT-1 [8] betrachtet. Dieser wird von nun an *ISet* genannt.

Der Algorithmus *ISet* berechnet ein „imaginäres“ VC C und fügt die übrig gebliebenen Knoten zum IS I hinzu.

Algorithmus 4.2.2 Algorithmus zur Berechnung eines Independent Set

```

1: procedure ISET( $G_b = (V_r, V_c, E), \rho$ )
2:   while  $E \neq \emptyset$  do
3:     if  $\Delta(V_r) > \rho \cdot \Delta(V_c)$  then
4:       Entferne alle Knoten  $v_r$  aus  $V_r$  mit  $d(v_r) = \Delta(V_r)$ 
5:       Entferne inzidente Kanten zu den Knoten  $v_r$ 
6:     else
7:       Entferne alle Knoten  $v_c$  aus  $V_c$  mit  $d(v_c) = \Delta(V_c)$ 
8:       Entferne inzidente Kanten zu den Knoten  $v_c$ 
9:     end if
10:  end while
11:  Füge die verbliebenen Knoten  $v \in V_r \cup V_c$  zum IS  $I$  hinzu
12:  return IS  $I$ 
13: end procedure

```

In jedem Schritt dieses Algorithmus *ISet* wird anhand der maximalen Knotengrade $\Delta(V_r)$ und $\Delta(V_c)$ der Knoten aus V_r und V_c entschieden, ob Knoten aus V_r oder V_c gelöscht werden (also ins „imaginäre“ VC C kommen). Die Entscheidung wird wie folgt gefällt: Wenn in Zeile 3 der maximale Knotengrad der Knoten aus V_r größer ist als der maximale Knotengrad der Knoten

aus V_c , der mit einem vorgegebenen Faktor ρ (Originalfaktor in ADMIT-1 ist zwei) multipliziert wird, dann werden alle Knoten aus V_r mit diesem Knotengrad ausgewählt. Anderenfalls werden alle Knoten aus V_c mit dem höchsten Knotengrad gewählt. Die ausgewählten Knoten werden in Zeile 4 bzw. 7 aus der entsprechenden Knotenmenge entfernt und deren inzidente Kanten gelöscht (Zeile 5 bzw. 8). Dieser Schritt wird wiederholt bis die Kantenmenge E leer ist. Die übrig gebliebenen Knoten bilden dann das IS I (Zeile 11).

Das Array *Farbe*, das für jeden Knoten ein Element enthält, in dem die Farbe gespeichert wird, wird mit -1 initialisiert. Nach der Berechnung des IS I werden alle darin enthaltenen Knoten im Array *Farbe* mit der Farbe 0 gefärbt. Abschließend färbt die Heuristik *StarBicoloring* (Alg. 4.2.3) die Knoten, die im VC C enthalten sind, so, dass die berechnete Färbung ein *Star Bicoloring* von G_b ist. Dazu weist der Algorithmus 4.2.3 den noch nicht gefärbten Knoten wie folgt Farben zu: In jedem Schritt i wird ein Knoten $v_i \in V_r \cup V_c$ ausgewählt, der noch nicht gefärbt ist (Zeilen 4 und 5). Dann werden in Zeile 6 alle Distanz-1 Nachbarn w betrachtet, die von der Funktion $N_1(v_i, G_b)$ zurückgeliefert werden.

Wenn ein solcher Knoten w mit der Farbe 0 oder noch nicht gefärbt ist, dann werden alle Distanz-1 Nachbarn x von w , die mit einer Farbe > 0 gefärbt sind, betrachtet und ihre Farben werden in Zeile 9 im Array *verboteneFarben* für den Knoten v_i gesperrt.

Wenn ein Knoten w aber mit einer Farbe > 0 gefärbt ist, dann werden alle Distanz-1 Nachbarn x dieses Knotens, die auch mit einer Farbe > 0 gefärbt sind, betrachtet. Von den Nachbarknoten x von w werden wiederum die Distanz-1 Nachbarn y mit $\text{Farbe}[y] > 0$ betrachtet. Wenn ein Knoten w dieselbe Farbe wie einer seiner Distanz-2 Nachbarn y hat und die Knoten w und y unterschiedliche Knoten sind, dann wird in Zeile 15 diese Farbe für v_i im Array *verboteneFarben* gesperrt. Wenn alle möglichen Knotenkombinationen überprüft worden sind, wird dem Knoten v_i in Zeile 21 die kleinste, nicht gesperrte Farbe zugewiesen. Am Ende wird in Zeile 24 die maximale Farbanzahl, die den Knoten aus der Knotenmenge V_r zugewiesen worden ist, zu allen Knoten aus V_c , die nicht mit der Farbe 0 gefärbt sind, hinzuaddiert, damit keine zwei Knoten der beiden Knotenmengen gleich gefärbt sind.

In [13, Abschnitt 5.3] wird die Abschätzung der Laufzeit des Alg. *StarBicoloringSchema* mit $O(|V_r \cup V_c| \cdot \bar{\delta}_3(V_r \cup V_c))$ unter der Bedingung angegeben, dass es einen Algorithmus für die Berechnung des IS I gibt, dessen Aufwand linear zu der Anzahl der Kanten ist.

4.2.2 Beschränkte zweiseitige Färbung

Die Heuristik *BeschränktesStarBicoloringSchema* (Alg. 4.2.4) ist eine Modifikation der Heuristik *StarBicoloringSchema* (Alg. 4.2.1). Dieser Algorithmus wird so auf die Menge E_R beschränkt, dass eine mit der modifizierten Heuristik berechnete Färbung ein *Star Bicoloring* von G_b *beschränkt auf E_R* (Def. 3.2.5) ist.

Aus einer resultierenden Färbung können zwei Seed-Matrizen S_r und S_c bestimmt werden,

Algorithmus 4.2.3 Heuristik zur Berechnung eines Star Bicoloring von G_b

```

1: procedure STARBICOLORING( $G_b = (V_r, V_c, E), \text{Farbe}$ )
2:   Sei  $v_1, v_2, \dots, v_{|V_r \cup V_c|}$  eine gegebene Reihenfolge von  $V_r \cup V_c$ 
3:   Initialisiere Array verboteneFarben mit 0
4:   for  $i := 1, \dots, |V_r \cup V_c|$  do
5:     if  $\text{Farbe}[v_i] \neq 0$  then
6:       for each  $w \in N_1(v_i, G_b)$  do
7:         if  $\text{Farbe}[w] \leq 0$  then
8:           for each  $x \in N_1(w, G_b)$  mit  $\text{Farbe}[x] > 0$  do
9:              $\text{verboteneFarben}[\text{Farbe}[x]] := i$ 
10:          end for
11:        else
12:          for each  $x \in N_1(w, G_b)$  mit  $\text{Farbe}[x] > 0$  do
13:            for each  $y \in N_1(x, G_b)$  mit  $\text{Farbe}[y] > 0$  und  $y \neq w$  do
14:              if  $\text{Farbe}[w] = \text{Farbe}[y]$  then
15:                 $\text{verboteneFarben}[\text{Farbe}[x]] := i$ 
16:              end if
17:            end for
18:          end for
19:        end if
20:      end for
21:       $\text{Farbe}[v_i] := \min\{j > 0 : \text{verboteneFarben}[j] \neq i\}$ 
22:    end if
23:  end for
24:  Addiere  $\max\{\text{Farbe}[v_r] : v_r \in V_r\}$  zu allen  $\text{Farbe}[v_c]$  mit  $v_c \in V_c$  und  $\text{Farbe}[v_c] > 0$ 
25:  return Farbe
26: end procedure

```

so dass alle benötigten Elemente der Jacobi-Matrix direkt und vollständig aus der komprimierten Jacobi-Matrix $\tilde{J} = S_r \cdot J \cdot S_c$ bestimmt werden können.

Algorithmus 4.2.4 Heuristik zur Berechnung eines beschränkten Star Bicoloring von G_b

procedure BESCHRÄNKTESSTARBICOLORINGSCHEMA($G_b = (V_r, V_c, E), E_R$)

1. Berechnung eines IS I beschränkt auf E_R
2. Knoten aus dem IS I werden mit Farbe 0 gefärbt
3. Färben der Knoten aus dem VC $C = (V_r \cup V_c) \setminus I$

end procedure

Zur Berechnung des IS I wird der Algorithmus *ISet* nicht mit der Kantenmenge E , sondern mit E_R aufgerufen, um ein IS I bzgl. dieser Teilmenge zu berechnen. Es ist ausreichend, nur einem Knoten einer Kante $(v, w) \in E_R$ eine Farbe $\neq 0$ zuzuweisen. Diese Anpassung ist wichtig, da ansonsten oftmals zu viele Kanten überdeckt werden und das IS I somit mehr Knoten enthalten könnte.

Es wird wieder das Array *Farbe* mit dem Wert -1 initialisiert. Dann wird allen Knoten aus dem IS I im Array *Farbe* die Farbe 0 zugewiesen. Anschließend müssen die noch nicht gefärbten Knoten so gefärbt werden, dass die Bedingungen des *Star Bicoloring von G_b beschränkt auf E_R* nicht verletzt werden. Dazu färbt der Alg. 4.2.5 die Knoten wie folgt: In jedem Schritt wird ein noch nicht gefärbter Knoten $v \in V_r \cup V_c$ betrachtet. Dann werden die Distanz-1 Nachbarn w des Knotens v besucht.

Wenn ein Knoten w noch nicht gefärbt ist oder ihm die Farbe 0 zugewiesen wurde, dann werden alle Pfade (v_i, w, x) betrachtet. Ist nun der Knoten x mit einer Farbe > 0 gefärbt und mindestens eine der beiden Kanten (v_i, w) oder (w, x) in der Menge E_R enthalten (Zeile 9), dann wird in Zeile 10 im Array *verboteneFarben* die Farbe des Knotens x für den Knoten v_i gesperrt.

Wenn ein Distanz-1 Nachbar w von v_i mit einer Farbe > 0 gefärbt ist, dann werden alle Pfade (v_i, w, x, y) betrachtet, wobei die Knoten x und y auch mit einer Farbe > 0 gefärbt sein müssen und die Kante $(w, x) \in E_R$ sein muss (Zeile 15). Wenn nun auf einem solchen Pfad die Knoten w und y unterschiedliche Knoten und gleich gefärbt sind, dann wird in Zeile 18 die Farbe des Knotens x für v_i gesperrt.

Zum Schluß wird in Zeile 28 – wie in Alg. 4.2.3 – die maximale Farbanzahl, die den Knoten aus der Knotenmenge V_r zugewiesen worden ist, zu allen Knoten aus V_c , die nicht mit der Farbe 0 gefärbt sind, hinzuaddiert.

Lemma 4.2.1. *Die Heuristik BeschränktesStarBicoloringSchema liefert als Färbung ein Star Bicoloring von G_b beschränkt auf E_R in $O(|V_r \cup V_c| \cdot \tilde{\delta}_3(V_c \cup V_r))$ zurück.*

Beweis. Korrektheit: Der Algorithmus *ISet* liefert ein korrektes IS I zurück, da diese Menge bzgl. der Menge E_R berechnet wird, und somit mindestens einer der beiden Knoten, die zu einer Kante aus E_R inzident sind, nicht im IS I ist.

Algorithmus 4.2.5 Heuristik zur Berechnung eines beschränkten Star Bicoloring

```

1: procedure BESCHRÄNKTESSTARBICOLORING( $G_b = (V_r, V_c, E), E_R, \text{Farbe}$ )
2:   Sei  $v_1, v_2, \dots, v_{|V_r \cup V_c|}$  eine gegebene Reihenfolge von  $V_r \cup V_c$ 
3:   Initialisiere Array verboteneFarben mit  $-1$ 
4:   for  $i := 1, \dots, |V_r \cup V_c|$  do
5:     if  $\text{Farbe}[v_i] \neq 0$  then
6:       for each  $w \in N_1(v_i, G_b)$  do
7:         if  $\text{Farbe}[w] \leq 0$  then
8:           for each  $x \in N_1(w, G_b)$  mit  $\text{Farbe}[x] > 0$  do
9:             if  $(v_i, w) \in E_R$  oder  $(w, x) \in E_R$  then
10:               $\text{verboteneFarben}[\text{Farbe}[x]] := i$ 
11:            end if
12:          end for
13:        else
14:          for each  $x \in N_1(w, G_b)$  mit  $\text{Farbe}[x] > 0$  do
15:            if  $(w, x) \in E_R$  then
16:              for each  $y \in N_1(x, G_b)$  mit  $\text{Farbe}[y] > 0$  und  $y \neq w$  do
17:                if  $\text{Farbe}[w] = \text{Farbe}[y]$  then
18:                   $\text{verboteneFarben}[\text{Farbe}[x]] := i$ 
19:                end if
20:              end for
21:            end if
22:          end for
23:        end if
24:      end for
25:       $\text{Farbe}[v_i] := \min\{j > 0 : \text{verboteneFarben}[j] \neq i\}$ 
26:    end if
27:  end for
28:  Addiere  $\max\{\text{Farbe}[v_r] : v_r \in V_r\}$  zu allen  $\text{Farbe}[v_c]$  mit  $v_c \in V_c$  und  $\text{Farbe}[v_c] > 0$ 
29: end procedure

```

Die Heuristik *BeschränktesStarBicoloring* färbt die Knoten aus V_r und V_c aus denselben Gründen unterschiedlich (außer bei der Farbe 0) wegen 1. Bedingung (B1) aus Def. 3.2.5 wie der Alg. *StarBicoloring*.

Von jeder Kante $(v_r, v_c) \in E_R$ muss wegen (B2) aus Def. 3.2.5 mindestens ein Knoten mit einer Farbe > 0 gefärbt sein. Das ist immer erfüllt, da einer der beiden Knoten im VC C enthalten ist und somit entsprechend gefärbt wird.

Für jeden Pfad (v, w, x) mit $(v, w) \in E_R$ oder $(w, x) \in E_R$ und $\Phi(w) \neq 0$ muss nach (B3) aus Def. 3.2.5 die Bedingung $\Phi(v) \neq \Phi(w)$ gelten. Dies wird erfüllt, weil eine bedingte Verzweigung eingefügt worden ist, die überprüft, ob eine der Kanten in der Kantenmenge E_R enthalten ist.

Für jeden Pfad (v, w, x, y) , dessen Knoten w, x mit einer Farbe > 0 gefärbt sind, muss wegen (B4) aus Def. 3.2.5 die Bedingung $\Phi(v) \neq \Phi(x)$ oder $\Phi(w) \neq \Phi(y)$ gelten. Dazu betrachtet der Algorithmus alle Pfade (v, w, x, y) , bei denen den Knoten w, x und y eine Farbe > 0 zugewiesen worden ist. Wenn $\Phi(w) = \Phi(y)$ gilt, dann wird die Farbe des Knotens x für den Knoten v gesperrt. Wenn für den Knoten y die Bedingung $\Phi(y) = 0$ gilt, dann kann v beliebig gefärbt werden, da $\Phi(w) > 0$ gilt. Somit wird diese Bedingung nie verletzt, da der Knoten v als letzter eines solchen Pfades immer entsprechend gefärbt werden kann.

Abschätzung: Die Abschätzung der Heuristik *StarBicoloringSchema* gilt auch hier, da nie mehr Knoten und Pfade überprüft werden müssen. \square

4.2.3 Vorsortierung

Gegeben sei ein bipartiter Graph $G_b = (V_r, V_c, E)$, der zweiseitig gefärbt werden soll. Bei der beschränkten, zweiseitigen Färbung sei zusätzlich die Kantenmenge E_R gegeben.

Die Algorithmen zur Vorsortierung sind im Abschnitt 4.1.3 vorgestellt worden. Dort wurden die Knoten vorsortiert bevor sie durch eine einseitige Färbungsheuristik gefärbt wurden. Vor dem Färben durch die Heuristiken *StarBicoloringSchema* und *BeschränktesStarBicoloringSchema* sollen die Knoten nun vorsortiert werden.

Bevor mit der Heuristik *StarBicoloringSchema* der bipartite Graph G_b eingefärbt wird, können die Knoten wie bei der einseitigen Färbung vorsortiert werden. Dazu können die beiden Knotenmengen V_r und V_c jeweils unabhängig mit dem *LFO*-, *SLO*- oder *IDO*-Algorithmus vorsortiert werden, wobei auf beide Knotenmengen derselbe Algorithmus angewendet wird.

Vor der Berechnung eines *Star Bicoloring* von G_b beschränkt auf E_R mit der Färbungsheuristik *BeschränktesStarBicoloringSchema* können die Knotenmengen V_r und V_c mit den Algorithmen *LFObeschränkt*, *SLObeschränkt* oder *IDObeschränkt* vorsortiert werden.

Nun kommen wir zu einem Beispiel, das aufzeigt, warum eine Vorsortierung auch beim zweiseitigen Färben sinnvoll ist: Seien alle Knoten einer Knotenmenge im IS I enthalten und die Knoten der anderen Knotenmenge im VC C enthalten. Dann werden die Knoten im VC C wie bei der entsprechenden einseitigen Färbungsheuristik gefärbt. Somit hat eine Vorsortierung

denselben Effekt wie bei den einseitigen Färbungsheuristiken. Dieses Beispiel kann auch für eine Komponente des Graphen G_b gelten, wenn dieser aus mehreren Zusammenhangskomponenten besteht.

5 Ergebnisse und Bewertung

In Kapitel 4 wurden verschiedene Graphfärbungs- und Vorsortierungsalgorithmen vorgestellt. Die Güte der Resultate dieser Algorithmen soll hier anhand verschiedener Testmatrizen verglichen werden. Dazu werden 24 Matrizen aus der Sammlung von Tim Davis [10] ausgewählt. Diese Matrizen sollen als Dünnbesetztheitsstrukturen von Jacobi-Matrizen betrachtet werden.

Die Matrizen werden in entsprechende bipartite Graphen $G_b = (V_r, V_c, E)$ überführt, um die Färbungsheuristiken anwenden zu können. Wie in Abschnitt 2.4 erklärt, repräsentiert ein solcher Graph die Dünnbesetztheitsstruktur einer Jacobi-Matrix J . Die Matrizen liegen in der Koordinatenform des Matrix Market Formats [2] vor. Alle in der Datenstruktur enthaltenen Elemente werden als Nichtnullelemente (NNE) betrachtet, auch wenn diese den Wert 0 haben. Alle nicht enthaltenen Elemente sind dementsprechend Nullelemente.

Im Abschnitt 5.1 *Vollständige Färbungen* werden die bipartiten Graphen $G_b = (V_r, V_c, E)$

- einseitig und vollständig mit der Heuristik *EinseitigeD2Färbung* und
- zweiseitig und vollständig mit der Heuristik *StarBicoloringSchema*

gefärbt. Zusätzlich werden die Knoten der bipartiten Graphen G_b vorsortiert und diejenige Färbungsheuristik angewendet, die ohne Vorsortierung insgesamt (für alle Matrizen) die geringste Farbanzahl zurückliefert.

Im darauffolgenden Abschnitt 5.2 sollen die Matrizen nur partiell berechnet werden. Die benötigten Elemente sind dabei die Hauptdiagonalelemente. Die entsprechenden bipartiten Graphen G_b werden

- einseitig und beschränkt mit der Heuristik *BeschränkteEinseitigeD2Färbung* und
- zweiseitig und beschränkt mit der Heuristik *BeschränktesStarBicoloringSchema*

gefärbt, wobei die Färbungen jeweils auf die entsprechende Kantenmenge E_R beschränkt sind, die die Kanten enthält, die den benötigten Elementen entsprechen. Nach dem Vorsortieren der Knoten, wird die Heuristik, die insgesamt die besten Ergebnisse zurückliefert, noch einmal auf die Graphen G_b angewendet.

5.1 Vollständige Färbungen

Die bipartiten Graphen $G_b = (V_r, V_c, E)$, die den Testmatrizen entsprechen, werden nun vollständig gefärbt. Dazu wird mit der Heuristik *EinseitigeD2Färbung* (Alg. 4.1.1) eine *einseitige Distanz-2 Färbung von G_b bzgl. V_c* und die gleiche Färbung bzgl. V_r berechnet. Zudem werden die Graphen G_b mit der Heuristik *StarBicoloringSchema* (Alg. 4.2.1) gefärbt. Mit dem Algorithmus, der insgesamt die beste Farbanzahl berechnet, werden die bipartiten Graphen noch einmal gefärbt, nachdem die Knoten mit den Algorithmen *LFO*, *SLO* und *IDO* vorsortiert worden sind (vgl. Kapitel 4).

Zuerst werden nun die Eigenschaften der Testmatrizen betrachtet, die in der Tabelle 5.1.1 aufgeführt sind. Dazu zählen die Anzahl der Spalten (bzw. Zeilen), die Anzahl der Nichtnullelemente, die strukturelle Symmetrie und der Grad der Dichtbesetztheit $\left(\frac{\#Nichtnullelemente}{\#Spalten^2}\right)$. Es ist zu erkennen, dass die Dimension der Matrizen im Bereich von 84.617×84.617 bis 213.360×213.360 liegt. Die Anzahl der Nichtnullelemente schwankt zwischen 463.625 und 11.063.545 Elementen. 10 Matrizen sind strukturell symmetrisch und 14 Matrizen dementsprechend nicht. Der Grad der Dichtbesetztheit liegt zwischen 0,0030 und 0,0735%.

In Tabelle 5.1.2 können die Farbanzahlen abgelesen werden, die die verschiedenen Heuristiken zum Färben der bipartiten Graphen G_b benötigen. Die Farbe 0 wird nicht mitgezählt, da sie für keinen AD-Berechnungsdurchlauf steht. Die 1. Spalte enthält die Namen der betrachteten Matrizen. In der 2. Spalte stehen die Farbanzahlen, die die Heuristik *EinseitigeD2Färbung* zum Färben der Spalten (Knotenmenge V_c) benötigt und in der 3. Spalte sind die Farbanzahlen derselben Heuristik bzgl. der Zeilen abzulesen. Die restlichen vier Spalten geben die Anzahl der Farben an, die die Heuristik *StarBicoloringSchema* benötigt, wobei dem Faktor ρ , der dem Algorithmus *ISet* übergeben wird, die Werte 1,0, 2,0, 3,0 und 3,5 zugewiesen worden sind.

Bei der einseitigen Färbung der Knotenmenge V_c werden insgesamt (für alle Matrizen) 70.207 Farben benötigt (70.207 AD-Berechnungsdurchläufe im Vorwärtsmodus (FM)). Wenn jedoch die Knotenmenge V_r einseitig gefärbt wird, dann werden nur 66.715 Farben benötigt. Allerdings müssen für diese Ersparnis von 3.492 Farben alle AD-Durchläufe im Rückwärtsmodus (RM) berechnet werden. Da wir wegen des geringeren Aufwands den FM bevorzugen, wird nun die Farbanzahl der einseitigen und vollständigen Färbung von V_c als Referenzgröße für die folgenden Betrachtungen gewählt.

Zudem ist anhand von Tabelle 5.1.2 zu erkennen, dass die Wahl des Faktors ρ bei dem Algorithmus *ISet* entscheidend ist. Bei der Wahl von $\rho = 1,0$ werden insgesamt 4.457 Farben benötigt. Im Gegensatz dazu werden bei $\rho = 3,0$ nur 2.694 Farben benötigt. Wenn der Faktor ρ den Wert 3,5 hat, dann werden insgesamt schon wieder mehr Farben (2.774) benötigt. Es gibt allerdings auch Matrizen, bei denen der Wert $\rho = 3,0$ keine gute Wahl ist. Ein Beispiel dafür ist die Matrix *para-5*, bei der der Faktor $\rho = 2,0$ zu einer Färbung mit weniger Farben führt.

Für die *einseitigen Distanz-2 Färbungen aller 24 Graphen bzgl. V_c* , die von der Heuris-

Matrix	#Spalten	#Nichtnullelemente	symmetrisch	dichtbesetzt (in %)
Ge87H76	112.985	7.892.195	1	0,0618
barrier2-10	115.625	3.897.557	0	0,0292
bmwcra_1	148.770	10.644.002	1	0,0481
c-73	169.422	1.279.274	1	0,0045
cage12	130.228	2.032.536	0	0,0120
cont-300	180.895	988.195	1	0,0030
d_pretok	182.730	1.641.672	1	0,0049
epb3	84.617	463.625	0	0,0065
ford2	100.196	544.688	1	0,0054
hcircuit	105.676	513.072	0	0,0046
lung2	109.460	492.564	0	0,0041
matrix_9	103.430	2.121.550	0	0,0198
ohne2	181.343	11.063.545	0	0,0336
para-5	155.924	5.416.358	0	0,0223
pkustk13	94.893	6.616.827	1	0,0735
rajat23	110.355	556.938	0	0,0046
scircuit	170.998	958.936	0	0,0033
ship_003	121.728	8.086.034	1	0,0546
shipsec1	140.874	7.813.404	1	0,0394
stomach	213.360	3.021.648	0	0,0066
torso2	115.967	1.033.473	0	0,0077
turon_m	189.924	1.690.876	1	0,0047
twotone	120.750	1.224.224	0	0,0084
xenon2	157.464	3.866.688	0	0,0156
$\Sigma =$	3.317.614	83.859.881		

Tabelle 5.1.1: 24 Matrizen aus der Tim Davis Matrizensammlung mit ihren charakteristischen Merkmalen: Spaltenanzahl, Anzahl der Nichtnullelemente (NNE), Symmetrie und Dichtbesetztheit in Prozent

tik *EinseitigeD2Färbung* zurückgeliefert werden, werden 70.207 Farben benötigt. Die Heuristik *StarBicoloringSchema* ($\rho = 3,0$) benötigt hingegen nur 2.694 Farben. Demnach werden bei der zweiseitigen Färbung nur ungefähr 4% der AD-Berechnungsdurchläufe der einseitigen Färbung benötigt. Bei der Matrix *c-73* führt die zweiseitige Färbung mit $\rho = 3,0$ im Vergleich zur einseitigen Färbung von V_c zur höchsten prozentualen Verringerung von Farben, so dass für die zweiseitige Färbung nur 0,14% der Farben der einseitigen Färbung benötigt werden (39.939 Farben zu 57 Farben). Die resultierende Farbanzahl der Heuristik *StarBicoloringSchema* ($\rho = 3,0$) ist allerdings bei 14 Matrizen nicht besser als die resultierende Farbanzahl der einseitigen Färbung der Knotenmenge V_c .

Die zweiseitige Färbung, bei der die Heuristik *StarBicoloringSchema* mit $\rho = 3,0$ aufgerufen wird, liefert insgesamt (für alle Matrizen) die Färbungen mit den wenigsten Farben zurück. In Tabelle 5.1.3 ist zu erkennen, dass bei diesen Färbungen den Knoten aus der Knotenmenge V_r

Matrix	EinseitigeD2Färbung bzgl.		StarBicoloringSchema mit				κ (in %)
	V_c	V_r	$\rho = 1,0$	$\rho = 2,0$	$\rho = 3,0$	$\rho = 3,5$	
Ge87H76	686	686	650	686	686	686	100,00
barrier2-10	8.437	8.437	807	118	118	118	1,40
bmwcra_1	453	453	195	186	186	261	41,06
c-73	39.939	39.939	57	57	57	57	0,14
cage12	96	96	96	96	96	96	100,00
cont-300	12	12	12	12	12	12	100,00
d_pretok	22	22	22	22	22	22	100,00
epb3	8	9	8	8	8	8	100,00
ford2	38	38	38	38	38	38	100,00
hcircuit	1.399	1.399	16	18	23	23	1,64
lung2	8	8	12	8	8	8	100,00
matrix_9	4.057	677	40	40	40	40	0,99
ohne2	3.441	3.441	169	168	204	240	5,93
para-5	6.931	6.931	532	132	162	162	2,34
pkustk13	303	303	264	303	303	303	100,00
rajat23	3.401	3.269	738	177	102	74	3,00
scircuit	353	353	172	106	86	83	24,36
ship_003	181	181	213	181	181	181	100,00
shipsec1	126	126	156	126	126	126	100,00
stomach	38	38	51	38	38	38	100,00
torso2	18	17	18	18	18	18	100,00
turon_m	23	23	23	23	23	23	100,00
twotone	185	205	116	104	105	105	56,76
xenon2	52	52	52	52	52	52	100,00
$\Sigma =$	70.207	66.715	4.457	2.717	2.694	2.774	

Tabelle 5.1.2: Farbanzahlen (ohne Farbe 0), die von den unterschiedlichen Färbungsheuristiken benötigt werden, und der Wert κ , der das Verhältnis der Farben der zweiseitigen Färbung zu denen der einseitigen Färbung ($\rho = 3,0$) von V_c prozentual angibt

169 Farben und den Knoten aus V_c 2.525 Farben (jeweils ohne Farbe 0) zugewiesen werden. Unter der Annahme, dass die Matrizen Dünnspartheitsstrukturen von Jacobi-Matrizen darstellen, würden also 169 AD-Berechnungsdurchläufe im Rückwärtsmodus (RM) und 2.525 im Vorwärtsmodus (FM) zur Berechnung der Werte benötigt.

Wenn davon ausgegangen wird, dass ein AD-Berechnungsdurchlauf im RM aufwendiger ist als einer im FM, dann muss nicht zwingend die Gesamtfarbanzahl minimiert werden, sondern der Mehraufwand für einen AD-Durchlauf im RM sollte berücksichtigt werden. Dazu kann man die Anzahl der benötigten Farben für die Zeilen mit einem Faktor multiplizieren, der den Mehraufwand widerspiegelt. Danach wird die Farbanzahl der Spalten hinzuaddiert. Wäre dieser Faktor 1,5, dann würden wir z.B. das *StarBicoloringSchema* mit $\rho = 4,5$, unabhängig von der Farbanzahl insgesamt, dem mit $\rho = 4,0$ vorziehen, da $112 * 1,5 + 2.687 = 2.855$ und

$101 * 1,5 + 2.691 = 2.842,5$ ist und somit die Färbung mit $\rho = 4,5$ zu einem geringeren AD-Berechnungsaufwand führt. Im Folgenden wird wieder die minimale Gesamtfarbanzahl betrachtet, ohne den Mehraufwand des RM zu berücksichtigen. Es fällt in Tabelle 5.1.3 vor allem auf, dass sich die Anzahl der Farben für die Zeilen immer weiter verringert, die Farbanzahl für die Spalten ab $\rho = 3,5$ aber zunimmt.

ρ	#Farben (alle Matrizen)	#Zeilenfarben	#Spaltenfarben
1,0	4.457	2.511	1.946
1,5	2.742	470	2.272
2,0	2.717	310	2.407
2,5	2.736	243	2.493
3,0	2.694	169	2.525
3,5	2.774	129	2.645
4,0	2.799	112	2.687
4,5	2.792	101	2.691
5,0	3.120	98	3.022
5,5	3.243	76	3.167

Tabelle 5.1.3: Färbung der 24 bipartiten Graphen durch das *StarBicoloringSchema* mit unterschiedlichen Faktoren ρ , wobei angegeben wird, wie viele Farben insgesamt, wie viele Farben für die Spalten und wie viele für die Zeilen benötigt werden

Mit der Heuristik *StarBicoloringSchema* ($\rho = 3,0$) sollen die bipartiten Graphen noch einmal gefärbt werden, nachdem die Knotenmengen V_c und V_r mit den Algorithmen *LFO*, *SLO* und *IDO* vorsortiert worden sind. In Tabelle 5.1.4 ist zu erkennen, dass bei der Vorsortierung mit Algorithmus *SLO* insgesamt die geringste Farbanzahl zum Färben benötigt wird. Im Gegensatz zu der Färbung, vor deren Anwendung die Knoten nicht vorsortiert wurden, können durch die Vorsortierung mit dem *SLO*-Algorithmus 239 AD-Berechnungsdurchläufe (oder ca. 9,61%) gespart werden. In der Tabelle kann zudem noch erkannt werden, dass *SLO* nicht für alle Matrizen die beste Vorsortierung ist. Bei der Matrix *Ge86H76* führen die Vorsortierungen mit *LFO* und *IDO* zu geringeren Farbanzahlen bei der anschließenden Färbung. Es kann sogar vorkommen, dass sich die Farbanzahl durch eine Vorsortierung erhöht. Ein Beispiel dafür ist der Fall, in dem die Matrix *bmwcra_1* mit *LFO* vorsortiert wird.

Die Tabelle 5.1.5 beinhaltet eine Übersicht der summierten Farbanzahlen aller Matrizen mit verschiedenen Verfahren, auf die in diesem Abschnitt eingegangen wurde. Die Heuristik *EinseitigeD2Färbung* benötigt 70.207 Farben. Wenn die Heuristik *StarBicoloringSchema* mit $\rho = 3,0$ aufgerufen wird, werden nur 3,84% der Farben benötigt. Wenn die bipartiten Graphen G_b zusätzlich noch mit *SLO* vorsortiert worden sind, dann werden sogar nur 3,47% der Farben benötigt. Somit können wir abschließend feststellen, dass für den in dieser Arbeit verwendeten Datensatz von Matrizen bei der vollständigen Färbung die Heuristik *StarBicoloringSchema* mit $\rho = 3,0$ zum Färben genutzt werden sollte, nachdem die Knoten mit dem Algorithmus *SLO* vorsortiert worden sind.

<i>StarBicoloringSchema</i> ($\rho = 3,0$)	–	<i>LFO</i>	<i>SLO</i>	<i>IDO</i>
Ge87H76	686	608	620	614
barrier2-10	118	118	118	118
bmwcra_1	186	195	138	141
c-73	57	57	57	59
cage12	96	72	68	70
cont-300	12	10	9	10
d_pretok	22	21	20	20
epb3	8	7	7	6
ford2	38	33	33	33
hcircuit	23	21	21	20
lung2	8	14	8	8
matrix_9	40	43	37	38
ohne2	204	204	204	204
para-5	162	162	162	162
pkustk13	303	300	300	300
rajat23	102	100	100	100
scircuit	86	85	85	85
ship_003	181	168	144	144
shipsec1	126	126	114	114
stomach	38	36	30	32
torso2	18	13	12	13
turon_m	23	22	20	20
twotone	105	84	87	89
xenon2	52	51	41	43
$\Sigma =$	2.694	2.550	2.435	2.443

Tabelle 5.1.4: Farbanzahlen der Färbungen bei Anwenden der Heuristik *StarBicoloringSchema* mit $\rho = 3,0$ ohne Vorsortierung und mit Vorsortierung durch *LFO*, *SLO* und *IDO*

Färbungsverfahren	Farbanzahl (insgesamt)
<i>EinseitigeD2Färbung</i>	70.207
<i>StarBicoloringSchema</i> ($\rho = 3,0$)	2.694
<i>StarBicoloringSchema</i> ($\rho = 3,0$) + <i>SLO</i>	2.435

Tabelle 5.1.5: Summierte Farbanzahlen mit den unterschiedlichen Färbungsheuristiken

5.2 Beschränkte Färbung

Nun sollen die Matrizen nicht mehr vollständig berechnet werden, sondern es müssen nur noch die Elemente der Hauptdiagonalen direkt bestimmt werden können. Die bipartiten Graphen $G_b = (V_r, V_c, E)$, die den 24 Testmatrizen des vorherigen Abschnitts entsprechen, werden mit der Heuristik *BeschränkteEinseitigeD2Färbung* (Alg. 4.1.2) einseitig bzgl. der Knotenmengen V_c und V_r gefärbt und mit der Heuristik *BeschränktesStarBicoloringSchema* (Alg. 4.2.4) zweiseitig gefärbt.

In Tabelle 5.2.1 werden zusätzliche Eigenschaften der Testmatrizen, die beim partiellen Berechnen der Hauptdiagonalen von Interesse sind, aufgeführt. Diese Merkmale sind die Anzahl der Nichtnullelemente auf der Hauptdiagonalen und die Angabe, wie viel Prozent der Hauptdiagonalen von Nichtnullelementen besetzt ist. Es ist zu erkennen, dass bei 17 Matrizen die Hauptdiagonale vollständig besetzt ist. Bei den restlichen Matrizen ist die Hauptdiagonale mit 42,54 bis 99,95% besetzt.

Matrix	#NNE auf HD	NNE auf HD (in %)
Ge87H76	112.985	100,00
barrier2-10	115.625	100,00
bmwcra_1	148.770	100,00
c-73	169.422	100,00
cage12	130.228	100,00
cont-300	90.597	50,08
d_pretok	129.160	70,68
epb3	84.617	100,00
ford2	100.196	100,00
hcircuit	105.628	99,95
lung2	109.460	100,00
matrix_9	103.430	100,00
ohne2	181.343	100,00
para-5	155.924	100,00
pkustk13	94.893	100,00
rajat23	106.832	96,80
scircuit	170.914	99,95
ship_003	121.728	100,00
shipsec1	140.874	100,00
stomach	213.360	100,00
torso2	115.967	100,00
turon_m	133.814	70,45
twotone	51.375	42,54
xenon2	157.464	100,00

Tabelle 5.2.1: Merkmale: Anzahl der Nichtnullelemente (NNE) auf der Hauptdiagonalen (HD) und prozentualer Anteil der NNE auf der HD

In Tabelle 5.2.2 ist zu erkennen, dass die Farbanzahlen der Matrizen bei der einseitigen Färbung der Knotenmenge V_c bzw. V_r mit der Heuristik *BeschränkteEinseitigeD2Färbung* und bei der zweiseitigen Färbung mit der Heuristik *BeschränktesStarBicoloringSchema* ($\rho = 1,0$) bis auf eine Ausnahme immer gleich sind.

Matrix	BeschränkteEinseitigeD2Färbung		BeschränktesStarBicoloringSchema
	V_c	V_r	$\rho = 1,0$
Ge87H76	224	224	224
barrier2-10	26	26	26
bmwcra_1	40	40	40
c-73	2	2	2
cage12	14	14	14
cont-300	2	2	1
d_pretok	6	6	6
epb3	5	5	5
ford2	28	28	28
hcircuit	5	5	5
lung2	4	4	4
matrix_9	7	7	7
ohne2	36	36	36
para-5	27	27	27
pkustk13	57	57	57
rajat23	9	9	9
scircuit	10	10	10
ship_003	60	60	60
shipsec1	48	48	48
stomach	8	8	8
torso2	5	5	5
turon_m	7	7	7
twotone	5	5	5
xenon2	20	20	20
$\Sigma =$	655	655	654

Tabelle 5.2.2: Anzahl der Farben (ohne Farbe 0), die von den unterschiedlichen Heuristiken zum beschränkten Färben der bipartiten Graphen G_b benötigt werden

Diese Ausnahme (Matrix *cont-300*) widerspricht nicht der Aussage von Lemma 3.2.12, die besagt, dass die minimalen Farbanzahlen der einseitigen und zweiseitigen Färbung bzgl. der benötigten Hauptdiagonalelemente gleich sein müssen. Zum einen wird bei dem Lemma eine vollbesetzte Hauptdiagonale vorausgesetzt. Diese Eigenschaft trifft hier allerdings nicht zu. Zum anderen liefern die Heuristiken nicht zwingend minimale Farbanzahlen zurück.

Ein solcher Fall, wie Matrix *cont-300*, kann z.B. auftreten, weil kein Knoten von der Heuristik *BeschränkteEinseitigeD2Färbung* mit der Farbe 0 gefärbt wird, obwohl die *einseitige Distanz-2 Färbung* von G_b bzgl. V_c beschränkt auf E_R dieses zuließe. In Abb. 5.2.1 sehen wir



Abbildung 5.2.1: Minimale Färbung in (a) im Gegensatz zu der Färbung der Heuristik *BeschränkteEinseitigeD2Färbung* in (b)

zwei gefärbte Beispielpfade. Die minimale Färbung ist in (a) zu sehen. Dort ist nur ein Knoten der Kante aus E_R mit einer Farbe $\neq 0$ gefärbt. In (b) ist die Färbung zu sehen, die die Heuristik *BeschränkteEinseitigeD2Färbung* berechnen würde. Bei einer entsprechenden Erweiterung dieses Algorithmus wäre es möglich, dass man dort dieselbe Farbanzahl wie bei der zweiseitigen Färbung erhält.

Wenn die Heuristik *BeschränktesStarBicoloringSchema* mit dem Faktor $\rho = 1,0$ aufgerufen wird, dann sind alle Knoten aus der Knotenmenge V_r im IS I enthalten. Somit ist kein Knoten aus V_r im VC C . Wenn dem Faktor ρ ein größerer Wert als 1.0 zugewiesen würde, dann käme wegen der Auswahlregel im Algorithmus *ISet* erst recht kein Knoten aus V_r ins VC C . Somit kann die Heuristik *BeschränkteEinseitigeD2Färbung* zur Färbung der bipartiten Graphen G_b der Heuristik *BeschränktesStarBicoloringSchema* vorgezogen werden, da die Farbanzahl insgesamt nur ein wenig schlechter ist, aber die Heuristik eine geringere Laufzeit hat und nie AD-Berechnungsdurchläufe im RM ausgeführt werden müssen (vgl. vorheriges Kapitel).

Aus diesem Grund werden die summierten Farbanzahlen der bipartiten Graphen verglichen, die die Heuristik *BeschränkteEinseitigeD2Färbung* zurückliefert, wenn die Knotenmenge V_c vor der Färbung nicht vorsortiert oder mit den Algorithmen *LFObeschränkt*, *SLObeschränkt* und *IDObeschränkt* vorsortiert wird. Tabelle 5.2.3 enthält die entsprechenden Werte. Wenn der Vorsortierungsalgorithmus *SLObeschränkt* eingesetzt wird, werden die wenigsten AD-Berechnungsdurchläufe (586) benötigt. Somit wird die Anzahl der benötigten Durchläufe im Vergleich zu der nicht vorsortierten Färbung um ungefähr ein Zehntel verringert. In Tabelle 5.2.3 ist zu erkennen, dass nicht für jede Matrix *SLObeschränkt* die beste Vorsortierung ist. Für die Matrix *torso2* ist es z.B. *IDObeschränkt*.

Nun wollen wir betrachten, wie viele AD-Berechnungsdurchläufe gespart werden können, wenn die Jacobi-Matrizen nur partiell (Hauptdiagonalelemente) und nicht vollständig berechnet werden müssen. In Tabelle 5.2.4 sind nochmal die Farbanzahlen (insgesamt) der vollständigen Berechnung aufgeführt. Hinzugekommen sind die Farbanzahlen, die in diesem Abschnitt verglichen wurden. Bei der beschränkten Färbung aller bipartiter Graphen mit der Heuristik *BeschränkteEinseitigeD2Färbung* sind 655 Farben zum Färben ausreichend. Im Vergleich zu der einseitigen und vollständigen Färbung von V_c werden hierfür nur 0,93% der Farben benötigt. Wenn zusätzlich noch mit *SLO* vorsortiert wird, dann werden sogar nur 586 Farben und gleich viele AD-Berechnungsdurchläufe zur partiellen Berechnung benötigt. Das bedeutet, dass sogar nur noch 0,83% der Farben im Vergleich zur einseitigen und vollständigen Färbung benötigt

<i>BeschränkteEinseitigeD2Färbung</i> (V_c)	–	<i>LFObeschr.</i>	<i>SLObeschr.</i>	<i>IDObeschr.</i>
Ge87H76	224	223	223	223
barrier2-10	26	27	23	24
bmwcra_1	40	41	37	35
c-73	2	2	2	2
cage12	14	13	11	12
cont-300	2	2	2	2
d_pretok	6	7	6	6
epb3	5	5	4	4
ford2	28	27	27	27
hcircuit	5	4	4	4
lung2	4	6	4	4
matrix_9	7	7	7	7
ohne2	36	35	34	33
para-5	27	28	23	23
pkustk13	57	42	42	42
rajat23	9	9	9	9
scircuit	10	8	7	7
ship_003	60	54	48	48
shipsec1	48	54	36	42
stomach	8	6	6	6
torso2	5	7	5	4
turon_m	7	8	6	7
twotone	5	4	5	4
xenon2	20	18	15	15
$\Sigma =$	655	637	586	590

Tabelle 5.2.3: Farbanzahlen der Färbungen beim Anwenden der Heuristik *BeschränkteEinseitigeD2Färbung* ohne Vorsortierung und mit Vorsortierung durch *LFObeschränkt*, *SLObeschränkt* und *IDObeschränkt*

werden.

Bei der partiellen Berechnung (bzgl. der Hauptdiagonalelemente) werden also nur ungefähr 30% der Berechnungsdurchläufe im Vergleich zur vollständigen und zweiseitigen Berechnung der Jacobi-Matrizen benötigt. Dabei können bei der partiellen Berechnung sogar alle AD-Berechnungsdurchläufe im FM berechnet werden.

Es wurde an dieser speziellen Menge der benötigten Elemente gezeigt, dass es sinnvoll sein kann, Jacobi-Matrizen partiell zu berechnen.

Färbungsverfahren	Farbanzahl (insgesamt)
<i>EinseitigeD2Färbung</i>	70.207
<i>StarBicoloringSchema</i> ($p = 3, 0$)	2.694
<i>StarBicoloringSchema</i> ($p = 3, 0$) + <i>SLO</i>	2.435
<i>BeschränkteEinseitigeD2Färbung</i>	655
<i>BeschränkteEinseitigeD2Färbung</i> + <i>SLO</i>	586

Tabelle 5.2.4: Summierte Farbanzahlen mit den unterschiedlichen Färbungsheuristiken (vollständig und beschränkt)

6 Zusammenfassung und Ausblick

Nach der Modellierung der Graphfärbungen und der zugehörigen Probleme für die vollständige und partielle Berechnung von Jacobi-Matrizen wurde in der vorliegenden Arbeit erstmals die Komplexität der vier Graphfärbungsprobleme untersucht, deren Färbungseigenschaften sich aus der Kombination einseitig oder zweiseitig und vollständig oder beschränkt zusammensetzen. Es wird in dieser Arbeit die NP-Vollständigkeit dieser vier Probleme bewiesen. Zudem wurde ein Sonderfall der beschränkten Färbung betrachtet, bei denen die Hauptdiagonalelemente partiell berechnet werden. Für diesen Sonderfall wurde gezeigt, dass die minimale Farbanzahl einer einseitigen und beschränkten Färbung identisch ist mit der minimalen Farbanzahl einer zweiseitigen Färbung. Für das Problem der einseitigen Färbung wurde die NP-Vollständigkeit gezeigt. Für das Problem der zweiseitigen Färbung wird die NP-Vollständigkeit vermutet.

Wegen der NP-Vollständigkeit der Probleme wurden in Kapitel 4 Heuristiken zur Einfärbung von bipartiten Graphen, die die Dünnbesetztheitsstrukturen von Jacobi-Matrizen darstellen, betrachtet. Zuerst wurden zwei existierende Heuristiken zur vollständigen Färbung dieser Graphen (einseitig und zweiseitig) vorgestellt. Diese Algorithmen wurden in der vorliegenden Arbeit erstmals so angepasst und implementiert, dass mit diesen jeweils eine beschränkte Färbung berechnet werden kann.

In Kapitel 5 wurde mit den vorgestellten Färbungsheuristiken für 24 Testmatrizen bzw. die passenden bipartiten Graphen die vier verschiedenen Färbungen berechnet, um die Güte der Algorithmen zu vergleichen. Bei der einseitigen und beschränkten Färbung für den Sonderfall der Hauptdiagonalelemente werden im Vergleich zur einseitigen und vollständigen Färbung nur noch ca. 1% der Farben (Linearkombinationen) benötigt und dennoch können alle benötigten Elemente direkt bestimmt werden. Anhand dieser Zahlen können wir feststellen, dass eine beschränkte Färbung bei diesen Testinstanzen sinnvoll ist, wenn nur die Hauptdiagonalelemente von J berechnet werden sollen. Durch eine zusätzliche Vorsortierung mit dem *SLO*-Algorithmus vor der Färbung reduziert sich die Anzahl der Linearkombinationen um ungefähr weitere 10%. Da in der Praxis ein oder mehrere Linearkombinationen von Spalten meistens aufwendiger zu berechnen sind als eine Vorsortierung der Knotenmenge, sollten die Knoten demnach vorsortiert werden.

Die zweiseitige und vollständige Färbung der 24 bipartiten Graphen benötigte im Vergleich zu der einseitigen und vollständigen Färbung bei der Berechnung nur ca. 3,8% der Linearkombinationen, wobei sich diese aus Linearkombinationen von Spalten und aus Linearkombinationen von Zeilen zusammensetzen. Mit einer zusätzlichen Vorsortierung verringert sich der Prozent-

satz werden weiter auf ca. 3,5%. Demnach ist eine vollständige Berechnung der Jacobi-Matrizen als Linearkombinationen von Spalten und Zeilen einer Berechnung, die nur aus Linearkombinationen aus Spalten besteht, vorzuziehen. Dieses trifft bei der partiellen Berechnung der Jacobi-Matrizen für den Sonderfall allerdings nicht zu, da für die Testmatrizen eine nahezu identische Anzahl an Linearkombinationen, die berechnet werden müssen, benötigt wird. Demnach ist die zweiseitige und beschränkte Färbung nicht von Interesse, wenn nur die Hauptdiagonale partiell berechnet werden soll.

Bei der partiellen Berechnung wurden in dieser Arbeit nur die Hauptdiagonalelemente als benötigte Elemente betrachtet, weil eine diagonale Vorkonditionierungsmatrix aus diesen Elementen konstruiert wird. Bei der partiellen Berechnung könnten weitere Mengen von benötigten Elementen betrachtet werden, die nicht nur Hauptdiagonalelemente, sondern auch andere Elemente enthalten. Ein Vorkonditionierer könnte z.B. so gewählt werden, dass dieser zusätzlich aus bestimmten Nebendiagonalelementen oder aber Blockdiagonalelementen besteht.

Des Weiteren könnten möglicherweise die Färbungsheuristiken verbessert werden. Bei der zweiseitigen Färbung könnte z.B. ein anderer Algorithmus zur Berechnung des Independent Sets genutzt werden, damit insgesamt weniger Knoten gefärbt werden müssen. Zudem könnte durch die Wahl einer anderen Färbungsstrategie die Anzahl der benötigten Farben verringert werden. Beim Automatischen Differenzieren ist das Berechnen einer Linearkombination aus Spalten aufwendiger als das Berechnen einer Linearkombination aus Zeilen. Vielleicht könnte dieser Aspekt der Gewichtung beim Färben berücksichtigt werden.

Zudem sollten vielleicht noch weitere Techniken zum Seeden, z.B. das Substitutions- und das Eliminationsverfahren, betrachtet werden. Dadurch könnten möglicherweise weitere Linearkombinationen eingespart werden. Allerdings muss hierbei die numerische Ungenauigkeit, die bei diesen Verfahren auftreten kann, berücksichtigt werden kann.

Literaturverzeichnis

- [1] BENZI, M.: *Preconditioning Techniques for Large Linear Systems: A Survey*. J. Comput. Phys., 182(2):418–477, 2002.
- [2] BOISVERT, R. F., R. POZO, and K. REMINGTON: *The Matrix Market Exchange Formats: Initial Design*. Technical Report NISTIR 5935, National Institute of Standards and Technology, Gaithersburg, MD, USA, December 1996.
- [3] BRÉLAZ, D.: *New Methods to Color the Vertices of a Graph*. Communications of the ACM, 22:251–256, 1979.
- [4] BÜCKER, M. und A. WOLF: *Beweisidee*. Mündliche Kommunikation, 2005.
- [5] COLEMAN, T. F. and J.-Y. CAI: *The Cyclic Coloring Problem and Estimation of Sparse Hessian Matrices*. SIAM J. Alg. Disc. Meth., 7(2):221–235, 1986.
- [6] COLEMAN, T. F. and J. J. MORÉ: *Estimation of Sparse Jacobian Matrices and Graph Coloring Problems*. SIAM J. Numer. Anal., 20(1):187–209, 1983.
- [7] COLEMAN, T. F. and A. VERMA: *The Efficient Computation of Sparse Jacobian Matrices Using Automatic Differentiation*. SIAM J. Sci. Comput., 19(4):1210–1233, 1998.
- [8] COLEMAN, T. F. and A. VERMA: *ADMIT-I: Automatic Differentiation and MATLAB Interface Toolbox*. ACM Trans. Math. Softw., 26(1):150–175, 2000.
- [9] CURTIS, A. R., M. J. D. POWELL, and J. K. REID: *On the Estimation of Sparse Jacobian Matrices*. Journal of the Institute of Mathematics and Applications, 13:117–119, 1974.
- [10] DAVIS, T.: *University of Florida Sparse Matrix Collection*. <http://www.cise.ufl.edu/research/sparse/matrices>. NA Digest, vol. 92, no. 42, October 16, 1994, NA Digest, vol. 96, no. 28, July 23, 1996, and NA Digest, vol. 97, no. 23, June 7, 1997.
- [11] GAREY, M. R. and D. S. JOHNSON: *Computers and Intractability: A Guide to the Theory of NP-Completeness*. Freeman, San Francisco, 1979.
- [12] GEBREMEDHIN, A. H.: *Practical Parallel Algorithms for Graph Coloring Problems in Numerical Optimization*. PhD thesis, University of Bergen, February 2003.

- [13] GEBREMEDHIN, A. H., F. MANNE, and A. POTHEN: *What Color Is Your Jacobian? Graph Coloring for Computing Derivatives*. SIAM Review, 47(4):629–705, 2005.
- [14] GRIEWANK, A.: *Evaluating Derivatives: Principles and Techniques of Algorithmic Differentiation*. Number 19 in *Frontiers in Appl. Math.* SIAM, Philadelphia, PA, 2000.
- [15] HOSSAIN, S. and T. STEIHAUG: *Computing a Sparse Jacobian Matrix by Rows and Columns*. Optimization Methods and Software, 10:33–48, 1998.
- [16] HOSSAIN, S. and T. STEIHAUG: *Reducing the Number of AD Passes for Computing a Sparse Jacobian Matrix*. In CORLISS, G., C. FAURE, A. GRIEWANK, L. HASCOËT, and U. NAUMANN (editors): *Automatic Differentiation of Algorithms: From Simulation to Optimization*, Computer and Information Science, chapter 31, pages 263–270. Springer, New York, NY, 2001.
- [17] HOSSAIN, S. and T. STEIHAUG: *Sparsity Issues in the Computation of Jacobian Matrices*. In *ISSAC '02: Proceedings of the 2002 International Symposium on Symbolic and Algebraic Computation*, pages 123–130, New York, NY, USA, 2002. ACM Press.
- [18] LIN, Y.-L. and S. S. SKIENA: *Algorithms for Square Roots of Graphs*. SIAM J. Discret. Math., 8(1):99–118, 1995.
- [19] MATULA, D., G. MARBLE, and J. ISAACSON: *Graph Coloring Algorithms*. In READ, R. (editor): *Graph Theory and Computing*, pages 109–122. Academic Press, New York, 1972.
- [20] RALL, L. B.: *Automatic Differentiation: Techniques and Applications*, volume 120 of *Lecture Notes in Computer Science*. Springer, Berlin, 1981.
- [21] SAAD, Y.: *Iterative Methods for Sparse Linear Systems*. PWS Publishing Company, Boston, 1996.
- [22] SCHWARZ, H.R. und N. KÖCKLER: *Numerische Mathematik*. Teubner, 2004.
- [23] SIEK, J. G., L.-Q. LEE, and A. LUMSDAINE: *The Boost Graph Library: User Guide and Reference Manual*. Addison-Wesley, 2002.
- [24] WELSH, D. J. A. and M. J. D. POWELL: *An upper bound for the chromatic number of a graph and its applications to timetabling problems*. Computer J., 10:85–87, 1967.